Business Case: Yulu - Hypothesis Testing

Introduction

Yulu is India's leading micro-mobility service provider, which offers unique vehicles for the daily commute. Starting off as a mission to eliminate traffic congestion in India, Yulu provides the safest commute solution through a user-friendly mobile app to enable shared, solo and sustainable commuting.

Yulu zones are located at all the appropriate locations (including metro stations, bus stands, office spaces, residential areas, corporate offices, etc) to make those first and last miles smooth, affordable, and convenient!

Yulu has recently suffered considerable dips in its revenues. They have contracted a consulting company to understand the factors on which the demand for these shared electric cycles depends. Specifically, they want to understand the factors affecting the demand for these shared electric cycles in the Indian market.

Problem Statement

The company wants to know:

1. Which variables are significant in predicting the demand for shared electric cycles in the Indian market?

2. How well those variables describe the electric cycle demands

In [704…
```
#importing all libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from scipy import stats
from scipy.stats import levene
from scipy.stats import norm, t
import statsmodels.api as sm
```

In [705…
```
import warnings
warnings.filterwarnings('ignore')
```

In [706…
```
#Loading the data
data = pd.read_csv('https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/
```

In [707…
```
#reading data
data.head(5)
```

Out[707]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0

In [708…

```
#shape of dataset
data.shape
```

Out[708]:

```
(10886, 12)
```

In [709…

```
#fields or attributes of dataset
data.columns
```

Out[709]:

```
Index(['datetime', 'season', 'holiday', 'workingday', 'weather', 'temp',
       'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count'],
      dtype='object')
```

In [710…

```
#basic information about dataset
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   datetime    10886 non-null  object 
 1   season      10886 non-null  int64  
 2   holiday     10886 non-null  int64  
 3   workingday  10886 non-null  int64  
 4   weather     10886 non-null  int64  
 5   temp        10886 non-null  float64
 6   atemp       10886 non-null  float64
 7   humidity    10886 non-null  int64  
 8   windspeed   10886 non-null  float64
 9   casual      10886 non-null  int64  
 10  registered  10886 non-null  int64  
 11  count       10886 non-null  int64  
dtypes: float64(3), int64(8), object(1)
memory usage: 1020.7+ KB
```

Season, holiday, working day, and weather are categorical/discrete variables, while the others are continuous variables.

In [711…

```
#checking missing values
data.isna().sum()
```

Out[711]:
```
datetime      0
season        0
holiday       0
workingday    0
weather       0
temp          0
atemp         0
humidity      0
windspeed     0
casual        0
registered    0
count         0
dtype: int64
```

In [712…
```
#double check of missing values
data.duplicated().sum()
```

Out[712]:
```
0
```

There are no missing values in dataset.

In [713…
```
# unique values in each column
data.nunique()
```

Out[713]:
```
datetime      10886
season            4
holiday           2
workingday        2
weather           4
temp             49
atemp            60
humidity         89
windspeed        28
casual          309
registered      731
count           822
dtype: int64
```

Statistical summary

In [714…
```
# Descriptive Statistics
data.describe().T

#casual and registered attributes might have outliers because their mean and media
#one another and the value of standard deviation is also high which tells us that
#in the data of these attributes.
```

Out[714]:

	count	mean	std	min	25%	50%	75%	max
season	10886.0	2.506614	1.116174	1.00	2.0000	3.000	4.0000	4.0000
holiday	10886.0	0.028569	0.166599	0.00	0.0000	0.000	0.0000	1.0000
workingday	10886.0	0.680875	0.466159	0.00	0.0000	1.000	1.0000	1.0000
weather	10886.0	1.418427	0.633839	1.00	1.0000	1.000	2.0000	4.0000
temp	10886.0	20.230860	7.791590	0.82	13.9400	20.500	26.2400	41.0000
atemp	10886.0	23.655084	8.474601	0.76	16.6650	24.240	31.0600	45.4550
humidity	10886.0	61.886460	19.245033	0.00	47.0000	62.000	77.0000	100.0000
windspeed	10886.0	12.799395	8.164537	0.00	7.0015	12.998	16.9979	56.9969
casual	10886.0	36.021955	49.960477	0.00	4.0000	17.000	49.0000	367.0000
registered	10886.0	155.552177	151.039033	0.00	36.0000	118.000	222.0000	886.0000
count	10886.0	191.574132	181.144454	1.00	42.0000	145.000	284.0000	977.0000

```
Datatype of following attributes needs to changed to proper data
type:
1. datetime - to datetime
2. season - to categorical
3. holiday - to categorical
4. workingday - to categorical
5. weather - to categorical
```

In [715…

```
# converting datetime from object to datetime
data['datetime'] = pd.to_datetime(data['datetime'])
```

In [716…

```
#statistical summary of columns containing categorical data
data_columns= ['season', 'holiday', 'workingday', 'weather']
for columns in data_columns:
    data[columns] = data[columns].astype('object')
data.describe(include = object).T
```

Out[716]:

	count	unique	top	freq
season	10886	4	4	2734
holiday	10886	2	0	10575
workingday	10886	2	1	7412
weather	10886	4	1	7192

Observations

1. 50% of the data has been recorded in the fall season, while 75% of the data has been recorded in the winter. Very few datapoints are recorded in the summer.
2. For the holiday feature, very few data points are collected for the holiday.
3. 50% of the data for the working day has been recorded, otherwise 75% of the data has been recorded during the cloudy season, while 50% is collected for a few clouds/clear.
4. The median temperature is noted at 20.5 degrees Celsius, while 75% of the data has been recorded at 26.24 degrees Celsius. The average temperature is noted as 20.36

degrees Celsius.

5. The median feeling temperature is 24.24 degrees Celsius, while 75% of the data has been recorded at 31.06 degrees Celsius. The average body temperature is noted as 23.65 degrees Celsius.
6. The median humidity is noted as 62, while 75% of the data has been recorded at 77. The average humidity is recorded at 61.86.
7. The median wind speed is noted as 12.998, while 75% of the wind speed is noted as 16.997. The average wind speed is noted as 12.79. The maximum humidity is 100.
8. The median causal users of the Yulu is 17, while 75% of the users are 49. The average #casual user is 36.02. The maximum number of causal users is 367.
9. The median number of registered users for the Yulu is 118, while 75% of the users are 222. The average number of registered users is 155.552. The maximum number of registered users is 886. 11. The median number of counted (casual + registered) users for the Yulu is 145, while 75% of the users are 284. The average number of counted users is 191.574. The maximum number of counted users is 977.

In [717…
```
data.describe().T
```

Out[717]:

	count	mean	std	min	25%	50%	75%	max
temp	10886.0	20.230860	7.791590	0.82	13.9400	20.500	26.2400	41.0000
atemp	10886.0	23.655084	8.474601	0.76	16.6650	24.240	31.0600	45.4550
humidity	10886.0	61.886460	19.245033	0.00	47.0000	62.000	77.0000	100.0000
windspeed	10886.0	12.799395	8.164537	0.00	7.0015	12.998	16.9979	56.9969
casual	10886.0	36.021955	49.960477	0.00	4.0000	17.000	49.0000	367.0000
registered	10886.0	155.552177	151.039033	0.00	36.0000	118.000	222.0000	886.0000
count	10886.0	191.574132	181.144454	1.00	42.0000	145.000	284.0000	977.0000

1. The average temperature was 20.23 degrees Celsius, with 20.5 happening 50% of the time.
2. The maximum temperature is recorded at 41, and the minimum temperature is recorded at 0.82 degrees Celsius.
3. The average feeling temperature was 23.65 degrees Celsius, with 24.24 happening 50% of the time.
4. The maximum felt temperature is recorded at 45.45, and the minimum temperature is recorded at 0.76 degrees Celsius.
5. The average wind speed was 12.8, with 12.998 happening 50% of the time.
6. The maximum wind speed was 56.996, and the minimum wind speed was recorded at 0.
7. The average humidity was 61.886%, with 62.8% of humidity happening 50% of the time.
8. The average number of casual users is 36.02, with 17 causal users being 50%.
9. The maximum number of casual users is 367, and the minimum number of users is zero.
10. The average number of registered users is 155.52, with 118 users 50% of the time.
11. The maximum and minimum number of registered users are 886 and 0, respectively.
12. The average number of counted users (causal + registered) is 191.57, with 145 customers 50% of the time.

13. The maximum and minimum number of total users are 977 and 1, respectively.

Non-Graphical Analysis: Value counts and unique attributes

In [718…

```
from IPython.display import display, HTML
CSS = """
.output {
    flex-direction: row;
}
"""
HTML('<style>{}</style>'.format(CSS))
```

Out[718]:

In [719…

```
# Unique values in each categorical columns

#Analysis data of season column
d1 = pd.DataFrame(data["season"].value_counts().reset_index())
d1.columns = ["season","Count"]
d1["% count"] = round((d1["Count"]/len(data))*100,2)

#Analysis data of weather column
d2 = pd.DataFrame(data["weather"].value_counts().reset_index())
d2.columns = ["weather","Count"]
d2["% count"] = round((d2["Count"]/len(data))*100,2)

display(d1)
display(d2)
```

	season	Count	% count
0	4	2734	25.11
1	2	2733	25.11
2	3	2733	25.11
3	1	2686	24.67

	weather	Count	% count
0	1	7192	66.07
1	2	2834	26.03
2	3	859	7.89
3	4	1	0.01

In [720…

```
#Analysis data of working day column
d3 = pd.DataFrame(data["workingday"].value_counts().reset_index())
d3.columns = ["working_day","Count"]
d3["% count"] = round((d3["Count"]/len(data))*100,2)

#Analysis data of holiday column
d4 = pd.DataFrame(data["holiday"].value_counts().reset_index())
d4.columns = ["holiday","Count"]
d4["% count"] = round((d4["Count"]/len(data))*100,2)

display(d3)
display(d4)
```

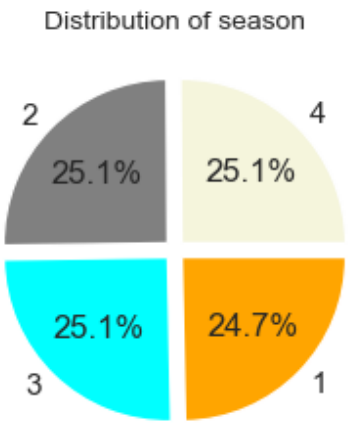
	working_day	Count	% count
0	1	7412	68.09
1	0	3474	31.91

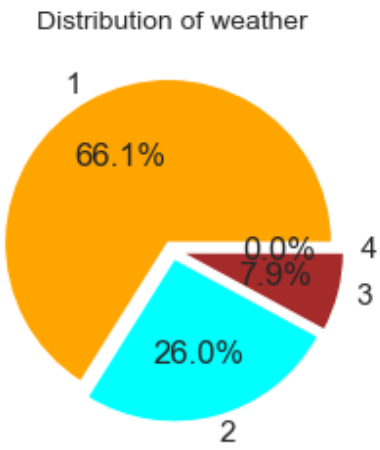
	holiday	Count	% count
0	0	10575	97.14
1	1	311	2.86

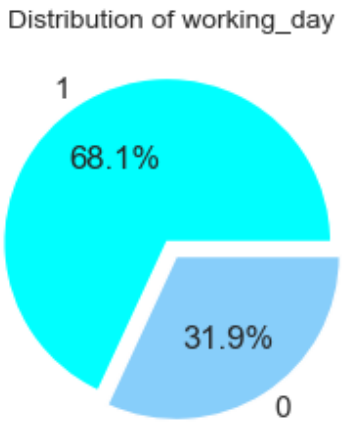
Percentage distribution over various fields

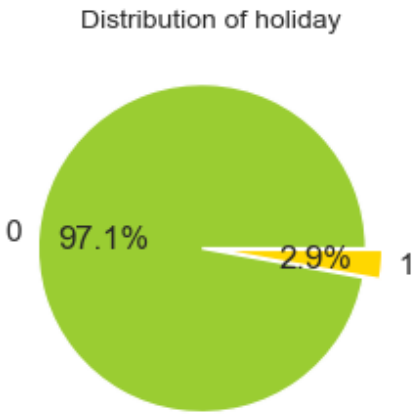
In [721…

```
plt.figure(figsize = (9,6))
explode_season = (0.06,0.06,0.06,0.06)
explode_weather = (0.05,0.05,0.05,0.05)
explode_holiday = (0.05,0.05)
explode_working_day = (0.05,0.05)
colors_season = ['beige',"grey", "cyan","orange"]
colors_weather = ["orange", "cyan", "brown","grey"]
colors_holiday = ['yellowgreen','gold']
colors_working_day = ["cyan", 'lightskyblue']

plt.subplot(2,2,1)
plt.pie(d1["Count"], labels=d1["season"], explode=explode_season,colors = colors_s
        autopct='%1.1f%%')
plt.title("Distribution of season", fontsize = 10)

plt.subplot(2,2,2)
plt.pie(d2["Count"], labels=d2["weather"], explode=explode_weather,colors = colors_
        autopct='%1.1f%%')
plt.title("Distribution of weather", fontsize = 10)

plt.subplot(2,2,3)
plt.pie(d3["Count"], labels=d3["working_day"], explode=explode_working_day,
        colors = colors_working_day, autopct='%1.1f%%')
plt.title("Distribution of working_day", fontsize = 10)

plt.subplot(2,2,4)
plt.pie(d4["Count"], labels=d4["holiday"], explode=explode_holiday,colors = colors_
        autopct='%1.1f%%')
plt.title("Distribution of holiday", fontsize = 10)

plt.show()
```









1. The datapoints collected for all the seasons are roughly equal in proportion.
2. The majority of the data is collected for non-holidays, which makes sense as a lot of people on holiday might be using the service.
3. 68% of the data points are collected for the working day considering that a lot of people use public transportation on working days.
4. Clear skies and Partly cloudy collects 66% of the data points, while mist or cloudy collects 26% of the data points.
5. Very few data points are collected during light snow or light rain conditions. Probably a lot of people don't use the service during heavy rain or thunder storms, which makes absolute sense.

Processing of data

In [722…

```
data['weekend']= np.where(((data['workingday']== 0) & (data['holiday']== 0)) , 1, (

data['weather']=data['weather'].replace([1,2,3,4],['Clear','Cloudy','Snow & Rain',
data["season"]=data["season"].replace([1,2,3,4],["spring","summer","fall","winter"
data["workingday"]=data["workingday"].replace([1,0],["Yes","No"])
data["holiday"]=data["holiday"].replace([1,0],["Yes","No"])

data["date"] = data["datetime"].dt.date

data["year"] = data["datetime"].dt.year
data["month"] = data["datetime"].dt.month
data["month_name"] = data["datetime"].dt.month_name()

data["day"] = data['datetime'].dt.day
```

```
data["day_name"] = data["datetime"].dt.day_name()
data["hour"] = data["datetime"].dt.hour
data['time'] = data['datetime'].dt.time
```

In [723…
```
#showing newly added fields for better analysis.
data.iloc[:, 9:].head(2)
```

Out[723]:

	casual	registered	count	weekend	date	year	month	month_name	day	day_name	hour
0	3	13	16	1	2011-01-01	2011	1	January	1	Saturday	0
1	8	32	40	1	2011-01-01	2011	1	January	1	Saturday	1

In [724…
```
# Checking for the minimum datetime and maximum datetime
data["datetime"].min(), data["datetime"].max()
```

Out[724]:
```
(Timestamp('2011-01-01 00:00:00'), Timestamp('2012-12-19 23:00:00'))
```

Conversion of some numerical datatype columns into categorical datatype

In [725…
```
def get_temp(temp):
    if temp <= 12:
        return "very low"
    elif temp > 12 and temp<24:
        return "low"
    elif temp >=24 and temp <35:
        return "moderate"
    elif temp >= 35:
        return "high"
data["temperature"] = pd.Series(map(get_temp,data["temp"]))
data["feeling_temp"] = pd.Series(map(get_temp,data["atemp"]))
data["windspeed_category"] = pd.qcut(data["windspeed"],8).astype("object")
```

In [726…
```
data.iloc[:, 11:].head(2)
```

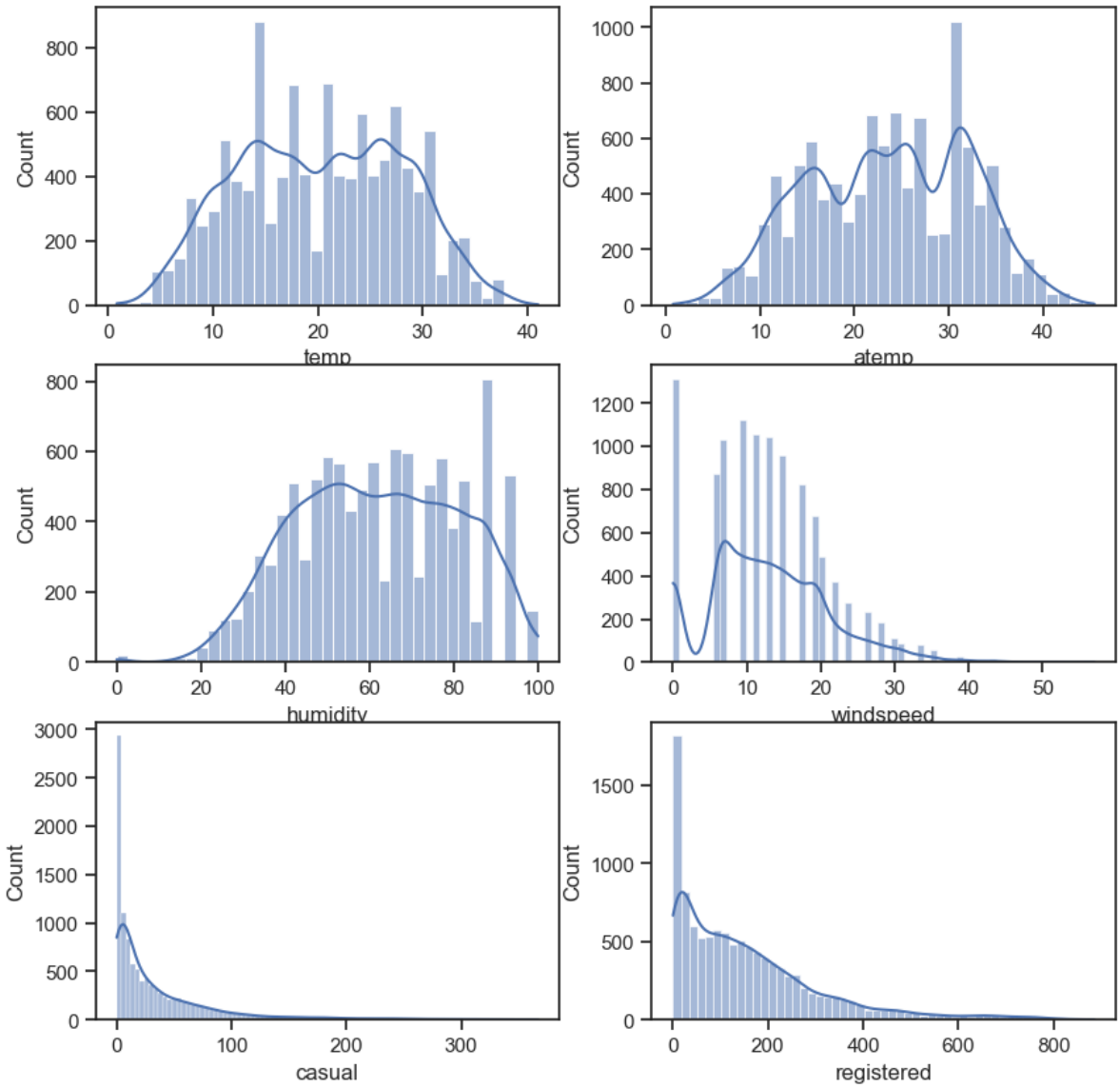
Out[726]:

	count	weekend	date	year	month	month_name	day	day_name	hour	time	temperat
0	16	1	2011-01-01	2011	1	January	1	Saturday	0	00:00:00	very
1	40	1	2011-01-01	2011	1	January	1	Saturday	1	01:00:00	very

Univariate Analysis

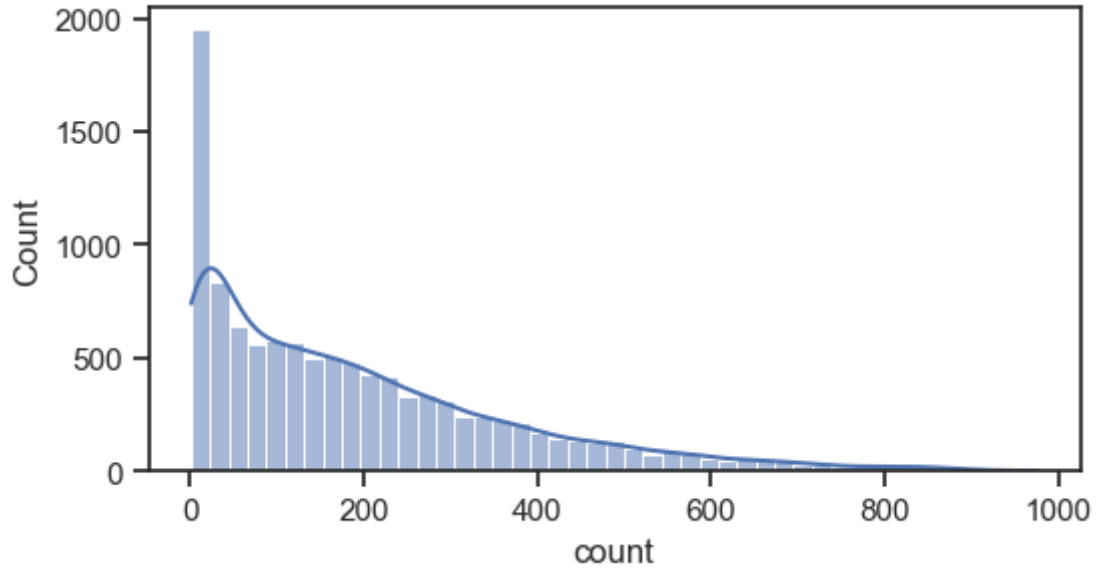
In [727…
```
#Univariate analysis of numerical fields
num_cols = ['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered','coun

fig, axis = plt.subplots(nrows=3, ncols=2, figsize=(10, 10))

index = 0
for row in range(3):
    for col in range(2):
        sns.histplot(data[num_cols[index]], ax=axis[row, col], kde=True)
```

```
        index += 1
plt.show()
```



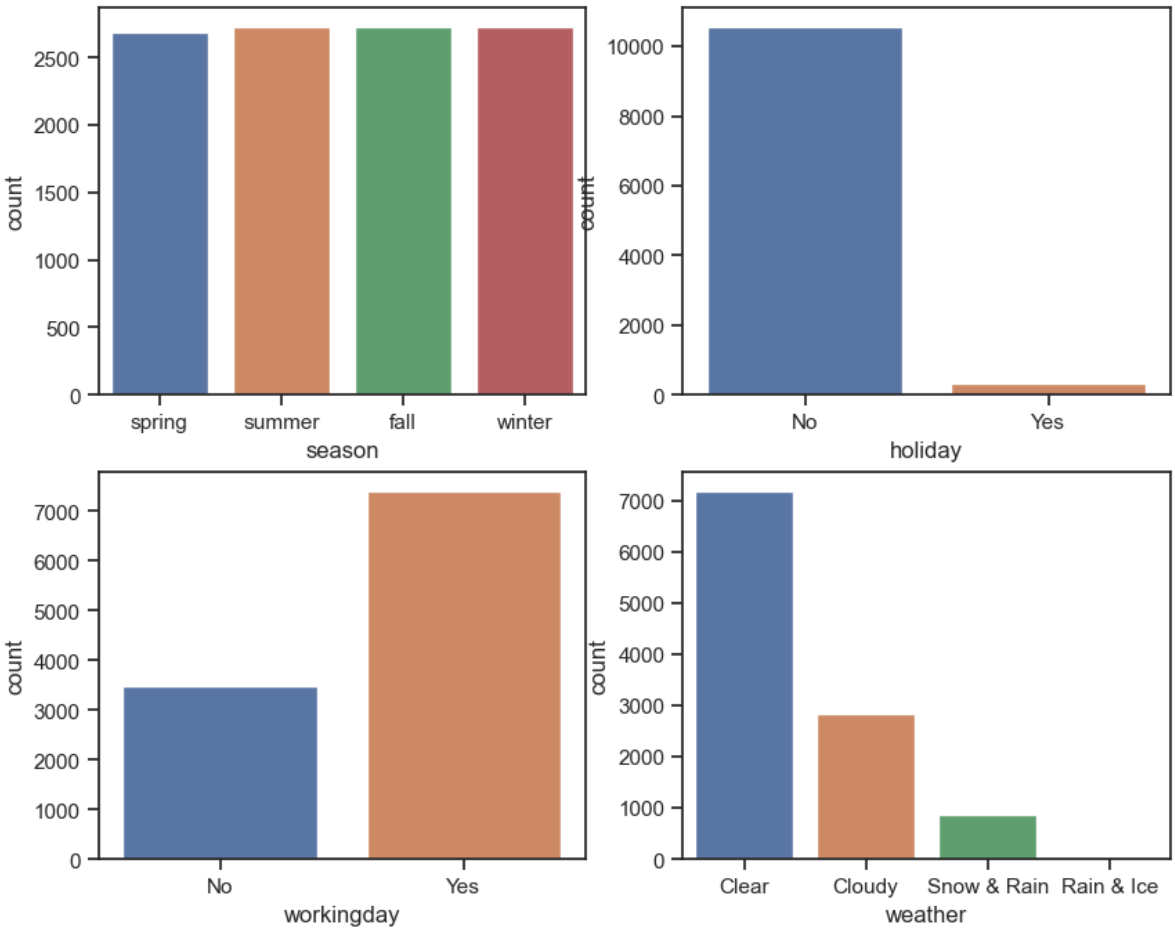
In [728…

```
plt.figure(figsize=(6,3))
sns.histplot(data[num_cols[-1]], kde=True)
plt.show()
```



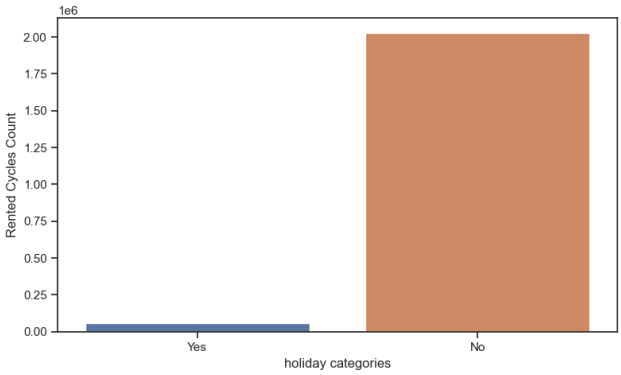
In [729…

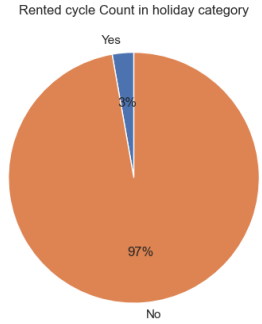
```
# countplot of each categorical column
cat_columns= ['season', 'holiday', 'workingday', 'weather']
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(10, 8))
index = 0
for row in range(2):
    for col in range(2):
        sns.countplot(data=data, x=cat_columns[index], ax=axis[row, col])
        index += 1
plt.show()
```

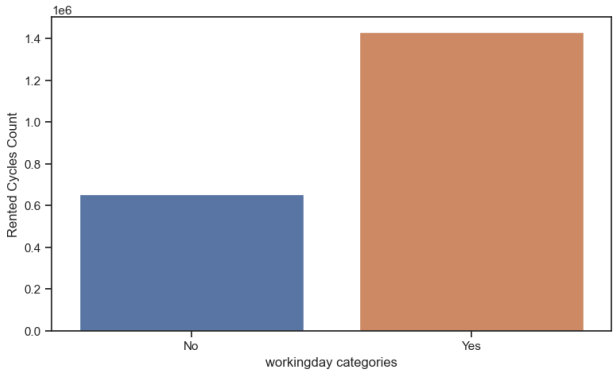


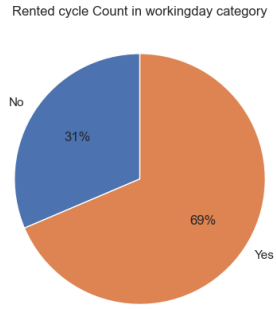
In [730…

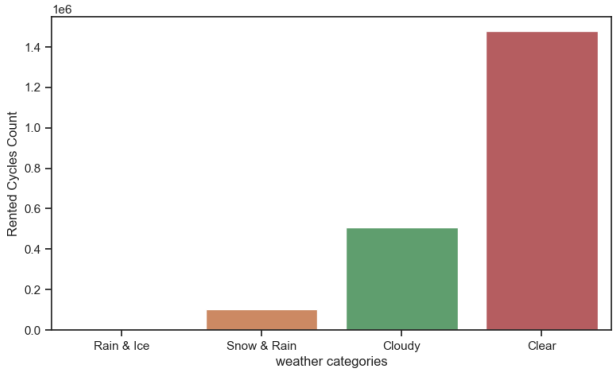
```
df = data
colnames=['holiday', 'workingday', 'weather', 'season', 'weekend', 'month', 'year'
plt.figure(figsize = (20, len(colnames)*6 ))
k=1
for colname in colnames:
  plt.subplot(len(colnames),2,k)
  s = df.groupby(colname)['count'].sum().sort_values(ascending=True)
  g = sns.barplot(data=s, x=s.index, y=s.values )
  g.set(xticklabels=s.index)
  g.set_ylabel("Rented Cycles Count")
  g.set_xlabel(f"{colname} categories")
  k+=1
  plt.subplot(len(colnames),2,k)
  plt.pie(s, labels=s.index, autopct='%.0f%%', startangle = 90)
  plt.title(f'Rented cycle Count in {colname} category')
  k+=1
```

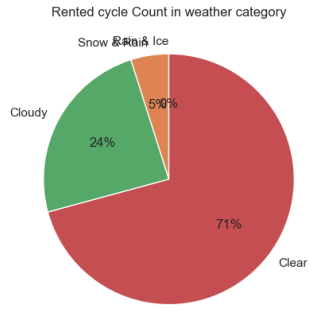

1e6
Rented Cycles Count
holiday categories
Yes
No

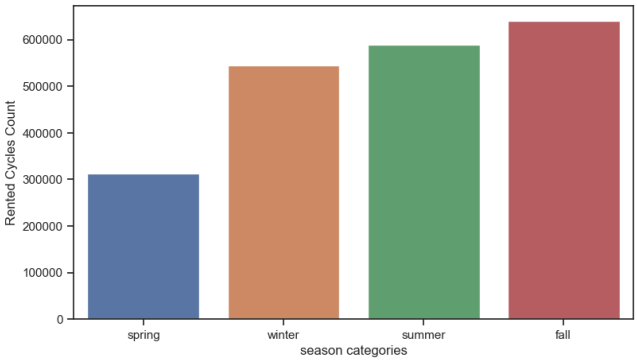

Rented cycle Count in holiday category
Yes
3%
97%
No

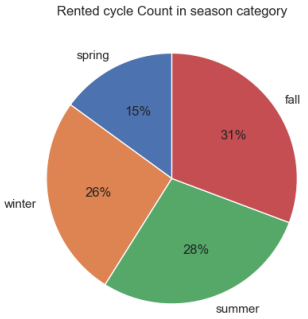

1e6
Rented Cycles Count
workingday categories
No
Yes

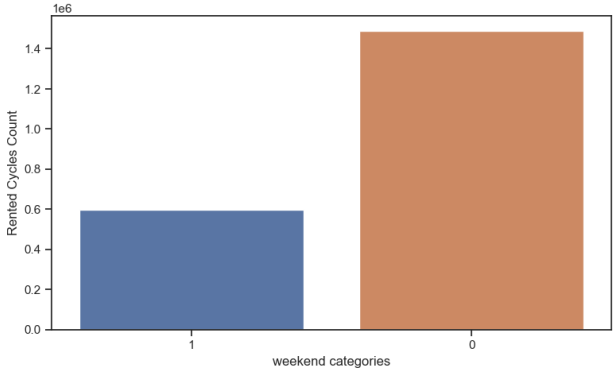

Rented cycle Count in workingday category
No
31%
69%
Yes

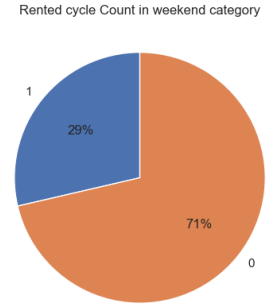

1e6
Rented Cycles Count
Rain & Ice
Snow & Rain
Cloudy
Clear
weather categories

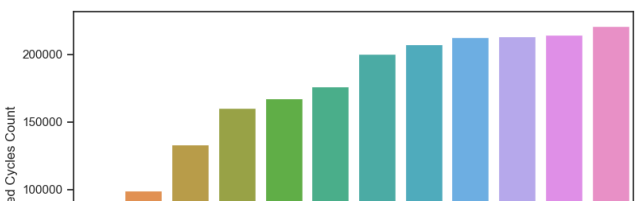

Rented cycle Count in weather category
Cloudy
24%
71%
Clear

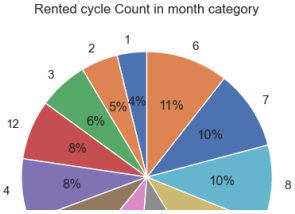

Rented Cycles Count
spring
winter
summer
fall
season categories


Rented cycle Count in season category
spring
15%
fall
31%
winter
26%
28%
summer


1e6
Rented Cycles Count
1
0
weekend categories


Rented cycle Count in weekend category
1
29%
71%
0


Cycles Count


Rented cycle Count in month category
1
2
3
4
5%
6%
8%
8%
10%
10%
11%
12

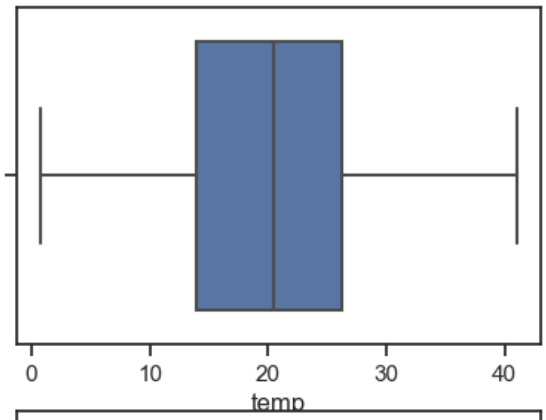
Univariate Analysis: Observations

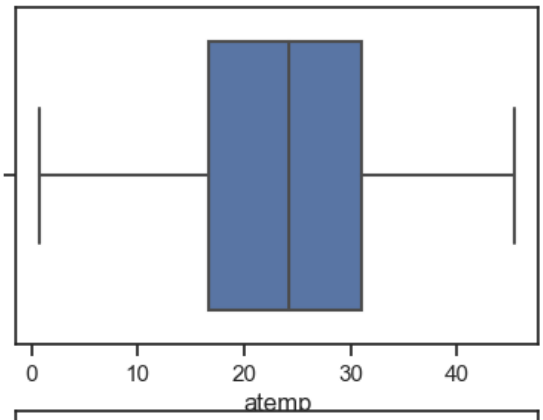
1. Season: Season 3 sees the highest amount of cycles rented, followed by 2, 4 and 1.
2. Weather: Users mostly rent cycle during weather 1, followed by 2 and 3. Weather 4 was recorded only once in the period of 2 years.
3. Atemp: The average temperature while users rent cycles fall between 17 degrees to 31 degrees.
4. Humidity: Humidity generally is between 47 and 77, with some outliers at 0.
5. Windspeed: The 25 percentile to 75 percentile windspeed is between 7km/he and 17km/hr.
6. Casual: average count of casual users on a hourly basis is 36 users. maximum casual users recorded in an hour is 367 and minimum is 0.
7. Registered: average count of registered users on an hourly basis is 155 users. maximum registered users in an hour is 886 and minimum is zero.
8. Count:
    - 2012 saw a significance rise in cycle rent count (63%) as compared to 2011 (37%)
    - only 21% cycle rents happened over the weekends , whereas weekdays contributed to 79% of cycle rent count.
    - Month of June (all at 11% of gross cycle rents) saw the maximam gross of Cycle rents, followed by May, July, August, September and October (all at 10% of gross cycle rents)
    - Gross Cycle rents Maximum at season 3 followed by Season 2.
    - Cycle rents were maximum at Weather Category 1 (71%), followed by weather category 2 (24%)

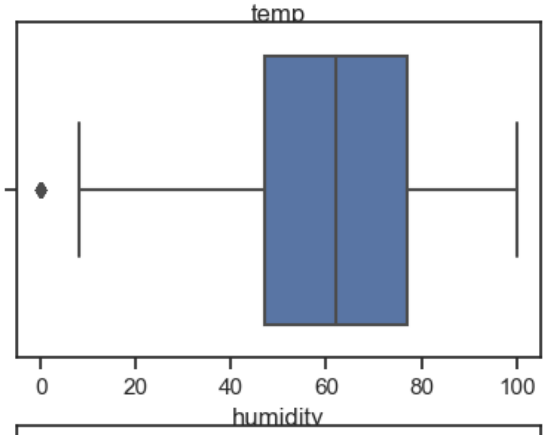
Outliers - visual representation

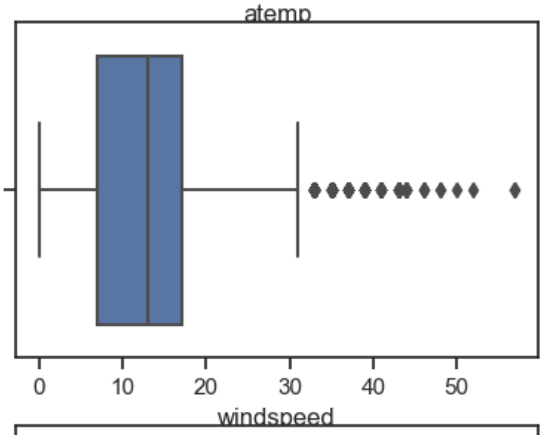
In [731…

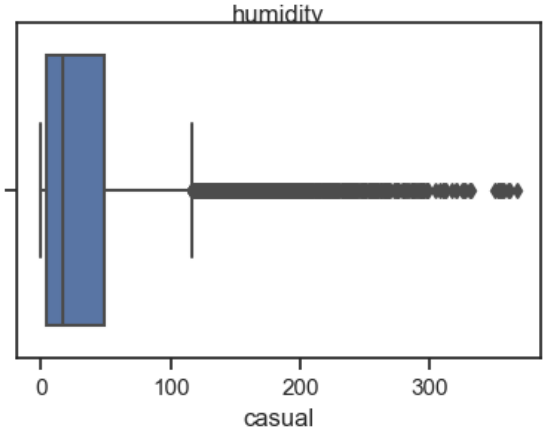
```
# plotting box plots to detect outliers in the data
num_cols = ['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered','coun
fig, axis = plt.subplots(nrows=3, ncols=2, figsize=(10, 10))
index = 0
for row in range(3):
    for col in range(2):
        sns.boxplot(x=data[num_cols[index]], ax=axis[row, col])
        index += 1
plt.show()
```

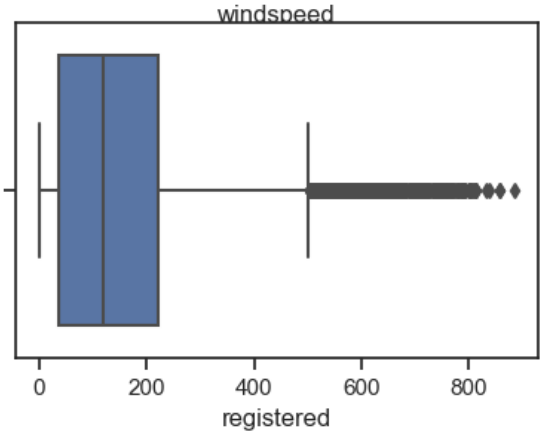






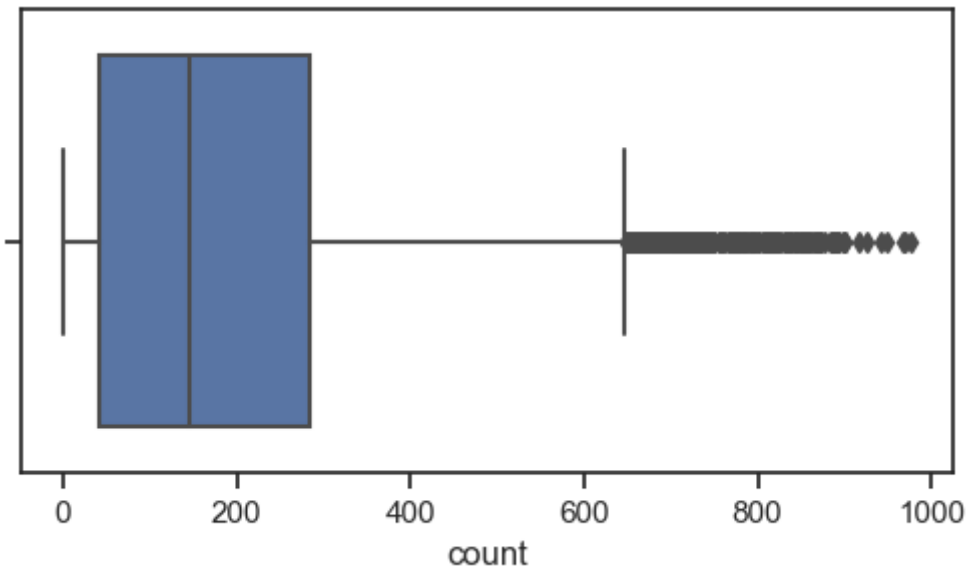






In [732…

```
plt.figure(figsize=(6,3))
sns.boxplot(x=data[num_cols[-1]])
plt.show()
```



Humidity, windspeed, casual, registered and count have outliers in the data.

Outliers - mathematical representation

In [733…

```
# Calculating IQR, upper and lower bound for count field in dataset
Q1_count = np.percentile(data["count"], 25,interpolation = 'midpoint')
Q3_count = np.percentile(data["count"], 75,interpolation = 'midpoint')
IQR_count = Q3_count - Q1_count

upper_count=Q3_count+1.5*IQR_count
lower_count=Q1_count-1.5*IQR_count

if lower_count < 0:
    lower_count =0

print("Upper Bound_count:",upper_count)
print("Lower Bound_count:",lower_count)

print("outliers above upper bound in 'count' field is:",
      len(data[data["count"]>=upper_count][["datetime","count"]]))
print("outliers below lower bound in 'count' field is:",
      len(data[data["count"]<=lower_count][["datetime","count"]]))
```

```
Upper Bound_count: 647.0
Lower Bound_count: 0
outliers above upper bound in 'count' field is: 303
outliers below lower bound in 'count' field is: 0
```

In [734…

```
# Calculating IQR, upper and lower bound for registered field in dataset
Q1_reg = np.percentile(data["registered"], 25,interpolation = 'midpoint')
Q3_reg = np.percentile(data["registered"], 75,interpolation = 'midpoint')
IQR_reg = Q3_reg - Q1_reg

upper_reg=Q3_reg+1.5*IQR_reg
lower_reg=Q1_reg-1.5*IQR_reg

if lower_reg < 0:
    lower_reg = 0

print("Upper Bound:",upper_reg)
print("Lower Bound:",lower_reg)

print("outliers above upper bound in 'registered' field is:",
      len(data[data["registered"]>=upper_reg][["datetime","registered"]]))
print("outliers below lower bound in 'registered' field is:",
      len(data[data["registered"]<=lower_reg][["datetime","registered"]]))
```

```
Upper Bound: 501.0
Lower Bound: 0
outliers above upper bound in 'registered' field is: 424
outliers below lower bound in 'registered' field is: 15
```

In [735…

```
# Calculating IQR, upper and lower bound for casual field in dataset
Q1_casual = np.percentile(data["casual"], 25,interpolation = 'midpoint')
Q3_casual = np.percentile(data["casual"], 75,interpolation = 'midpoint')
IQR_casual = Q3_casual - Q1_casual

upper_casual=Q3_casual+1.5*IQR_casual
lower_casual=Q1_casual-1.5*IQR_casual
```

```
if lower_casual < 0:
    lower_casual = 0

print("Upper Bound:",upper_casual)
print("Lower Bound:",lower_casual)

print("outliers above upper bound in 'casual' field is:",
      len(data[data["casual"]>=upper_casual][["datetime","casual"]]))
print("outliers below lower bound in 'casual' field is:",
      len(data[data["casual"]<=lower_casual][["datetime","casual"]]))
```

```
Upper Bound: 116.5
Lower Bound: 0
outliers above upper bound in 'casual' field is: 749
outliers below lower bound in 'casual' field is: 986
```

In [736…

```
# Calculating IQR, upper and lower bound for windspeed field in dataset
Q1_wind = np.percentile(data["windspeed"], 25,interpolation = 'midpoint')
Q3_wind = np.percentile(data["windspeed"], 75,interpolation = 'midpoint')
IQR_wind = Q3_wind - Q1_wind

upper_wind=Q3_wind+1.5*IQR_wind
lower_wind=Q1_wind-1.5*IQR_wind

if lower_wind < 0:
    lower_wind = 0

print("Upper Bound:",upper_wind)
print("Lower Bound:",lower_wind)

print("outliers above upper bound in 'windspeed' field is:",
      len(data[data["windspeed"]>=upper_wind][["datetime","windspeed"]]))
print("outliers below lower bound in 'windspeed' field is:",
      len(data[data["windspeed"]<=lower_wind][["datetime","windspeed"]]))
```

```
Upper Bound: 31.992500000000003
Lower Bound: 0
outliers above upper bound in 'windspeed' field is: 227
outliers below lower bound in 'windspeed' field is: 1313
```

In [737…

```
# Calculating IQR, upper and lower bound for humidity field in dataset
Q1_humidity = np.percentile(data["humidity"], 25,interpolation = 'midpoint')
Q3_humidity = np.percentile(data["humidity"], 75,interpolation = 'midpoint')
IQR_humidity = Q3_humidity - Q1_humidity

upper_humidity=Q3_humidity+1.5*IQR_humidity
lower_humidity=Q1_humidity-1.5*IQR_humidity

if lower_humidity < 0:
    lower_humidity = 0

print("Upper Bound:",upper_humidity)
print("Lower Bound:",lower_humidity)

print("outliers above upper bound in 'humidity' field is:",
      len(data[data["humidity"]>=upper_humidity][["datetime","humidity"]]))
print("outliers below lower bound in 'humidity' field is:",
      len(data[data["humidity"]<=lower_humidity][["datetime","humidity"]]))
```

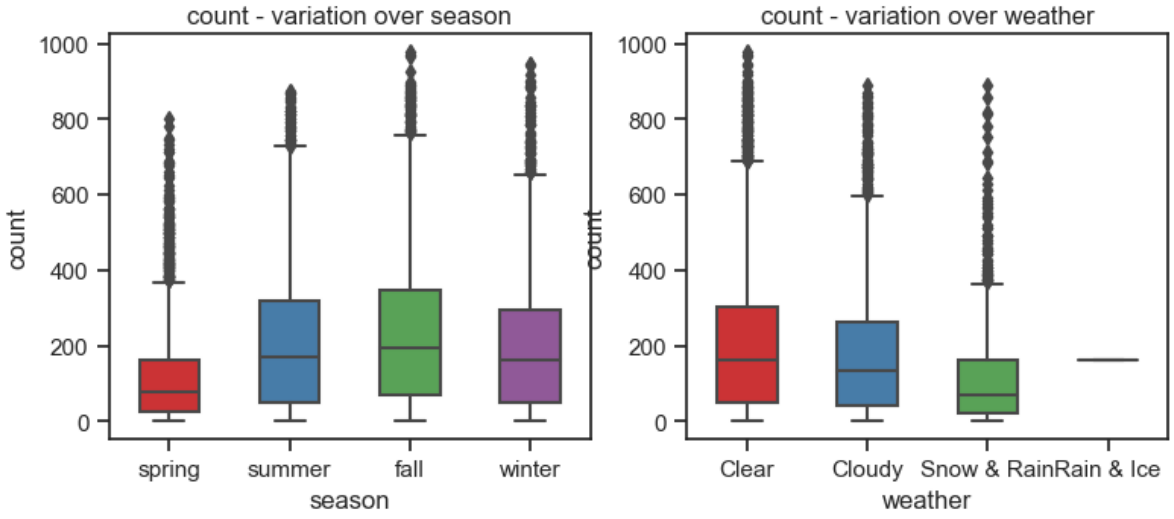
```
Upper Bound: 122.0
Lower Bound: 2.0
outliers above upper bound in 'humidity' field is: 0
outliers below lower bound in 'humidity' field is: 22
```

Bivariate Analysis

Distribution of "count" field with respect to categorical fields

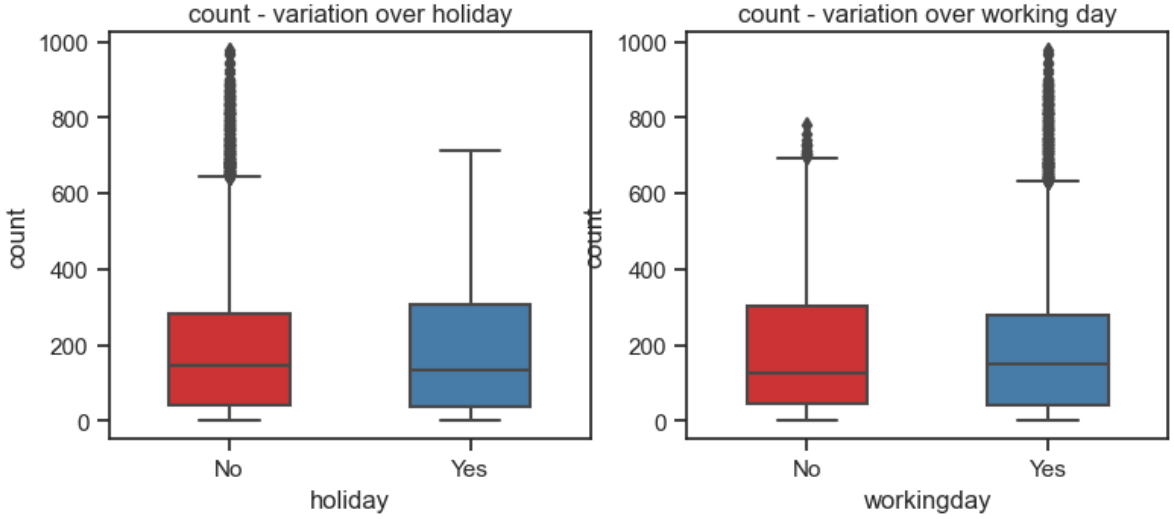
In [738…

```
#Outliers in count field with respect to season and weather
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(9, 3))
fig.subplots_adjust(top=1)

sns.boxplot(x=data["season"], y = data['count'], palette='Set1',width=0.5,ax=axis[(
axis[0].set_title("count - variation over season", pad=5, fontsize=12)

sns.boxplot(x=data["weather"], y = data['count'], palette='Set1',width=0.5, ax=axis
axis[1].set_title("count - variation over weather", pad=5, fontsize=12)
plt.show()
```



In [739…

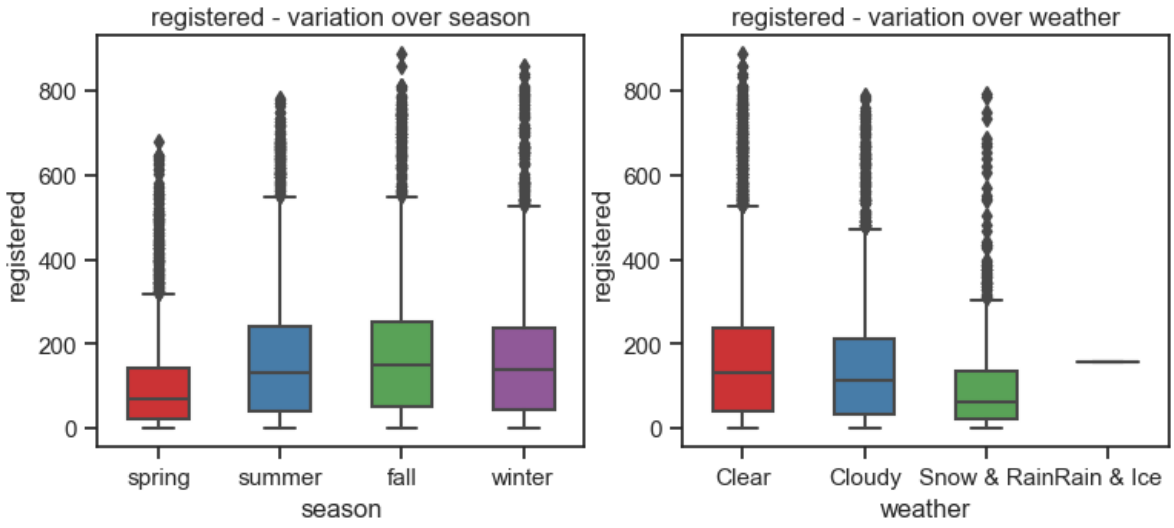
```
#Outliers in count field with respect to holiday and working day
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(9, 3))
fig.subplots_adjust(top=1)

sns.boxplot(x=data["holiday"], y = data['count'], palette='Set1',width=0.5,ax=axis
axis[0].set_title("count - variation over holiday", pad=5, fontsize=12)

sns.boxplot(x=data["workingday"], y = data['count'], palette='Set1',width=0.5, ax=
axis[1].set_title("count - variation over working day", pad=5, fontsize=12)
plt.show()
```



Distribution of "registered" field with respect to categorical fields

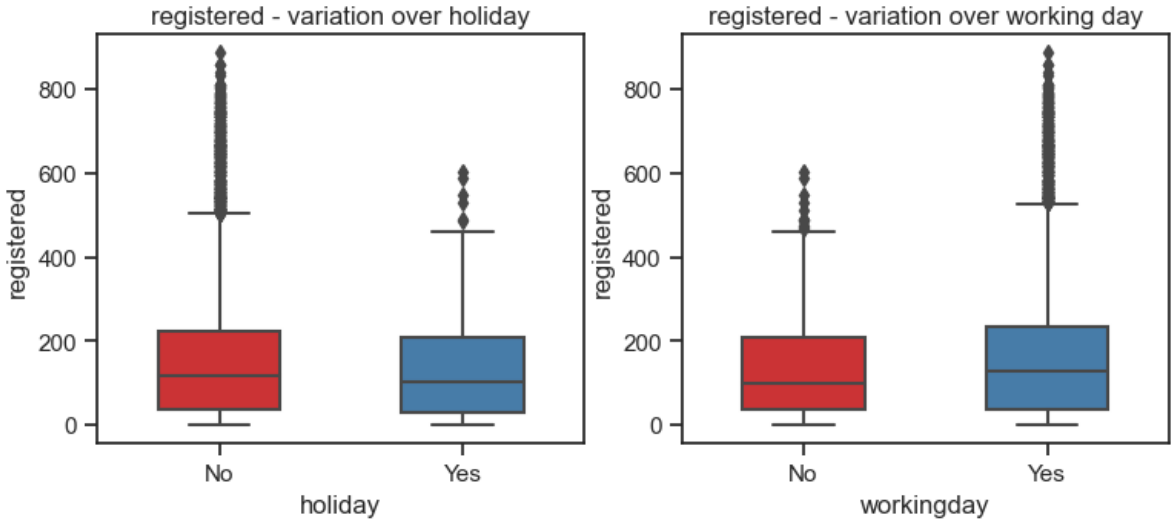
In [740…

```
#Outliers in registered field with respect to season and weather
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(9, 3))
fig.subplots_adjust(top=1)

sns.boxplot(x=data["season"], y = data['registered'], palette='Set1',width=0.5,ax=
axis[0].set_title("registered - variation over season", pad=5, fontsize=12)

sns.boxplot(x=data["weather"], y = data['registered'], palette='Set1',width=0.5, a
axis[1].set_title("registered - variation over weather", pad=5, fontsize=12)
plt.show()
```



In [741…

```
#Outliers in registered field with respect to holiday and working day
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(9, 3))
fig.subplots_adjust(top=1)

sns.boxplot(x=data["holiday"], y = data['registered'], palette='Set1',width=0.5,ax
axis[0].set_title("registered - variation over holiday", pad=5, fontsize=12)

sns.boxplot(x=data["workingday"], y = data['registered'], palette='Set1',width=0.5
axis[1].set_title("registered - variation over working day", pad=5, fontsize=12)
plt.show()
```

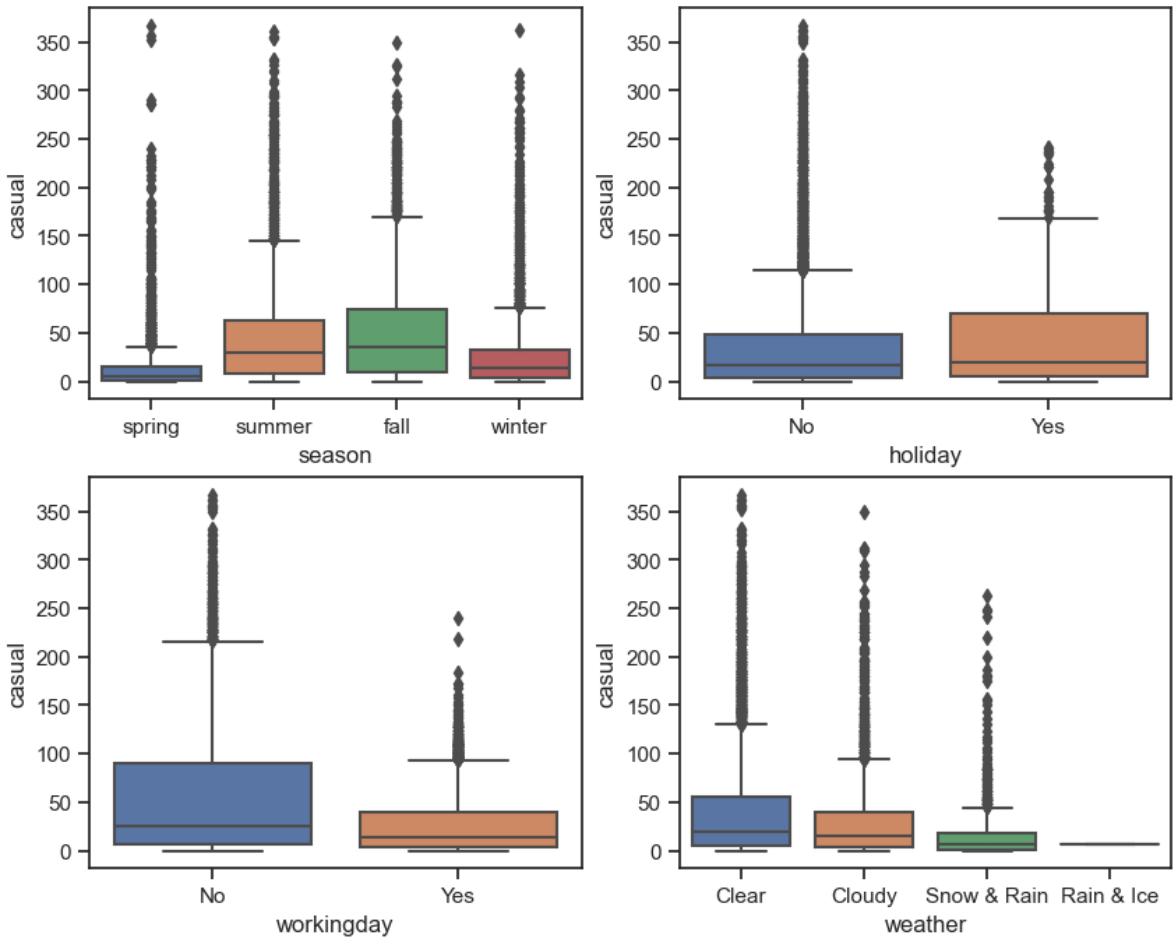


Distribution of "casual" field with respect to categorical fields

In [742…

```
cat_columns= ['season', 'holiday', 'workingday', 'weather']
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(10, 8))
```

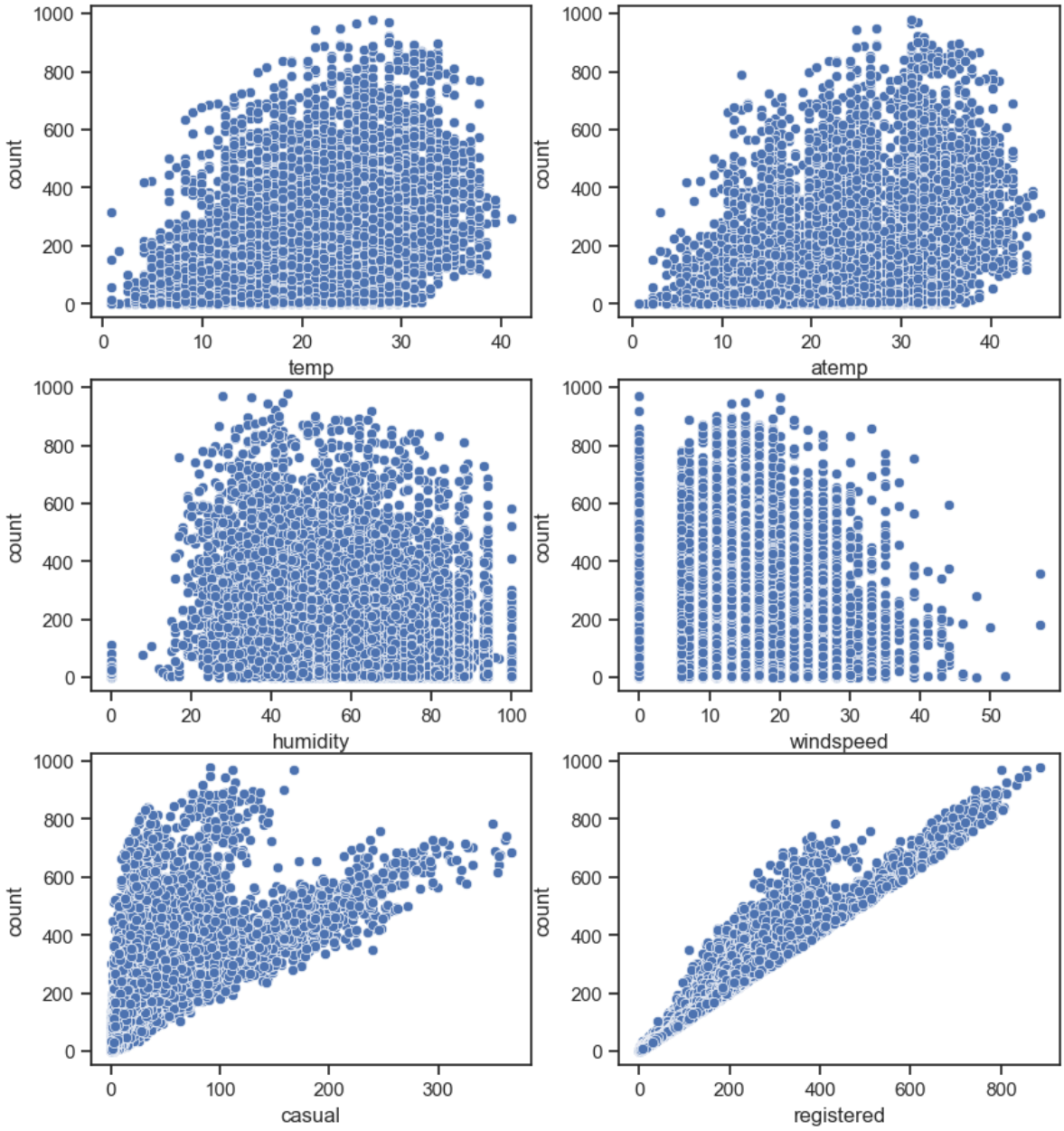
```
index = 0
for row in range(2):
    for col in range(2):
        sns.boxplot(data=data, x=cat_columns[index],y=data["casual"],ax=axis[row,
        index += 1
plt.show()
```



Distribution of numerical fields over "count" field

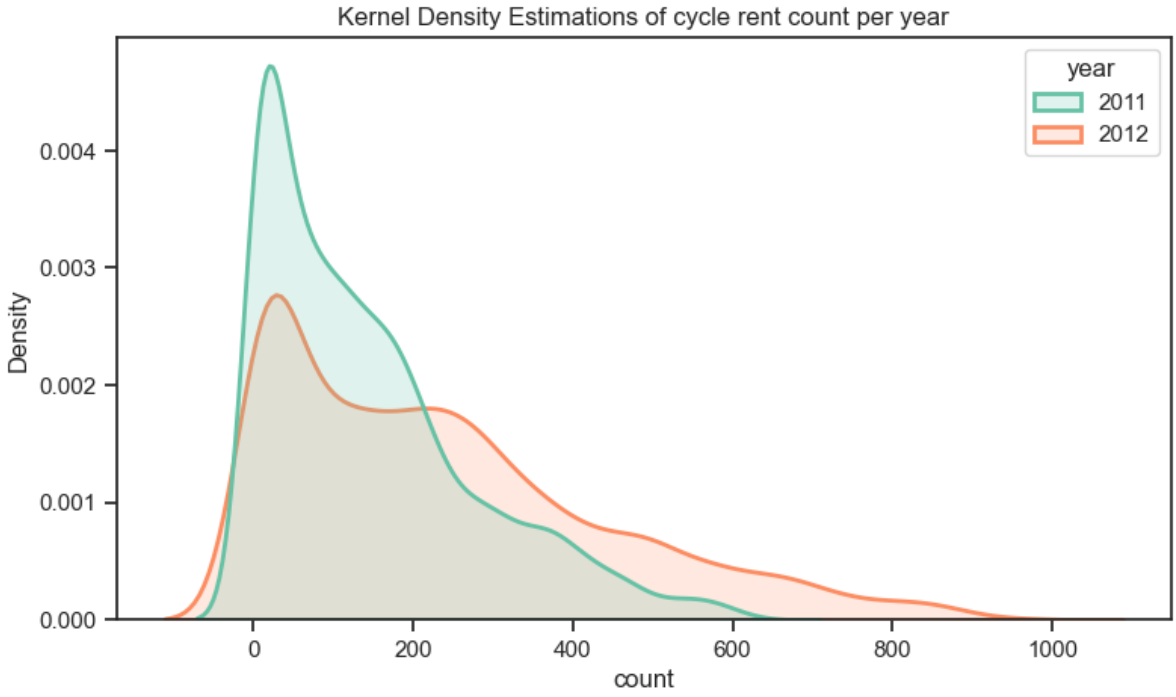
In [743…

```
num_columns= ['temp', 'atemp', 'humidity', 'windspeed','casual','registered']
fig, axis = plt.subplots(nrows=3, ncols=2, figsize=(10, 11))

index = 0
for row in range(3):
    for col in range(2):
        sns.scatterplot(data=data, x=num_columns[index], y='count',ax=axis[row, co
        index += 1
plt.show()
```



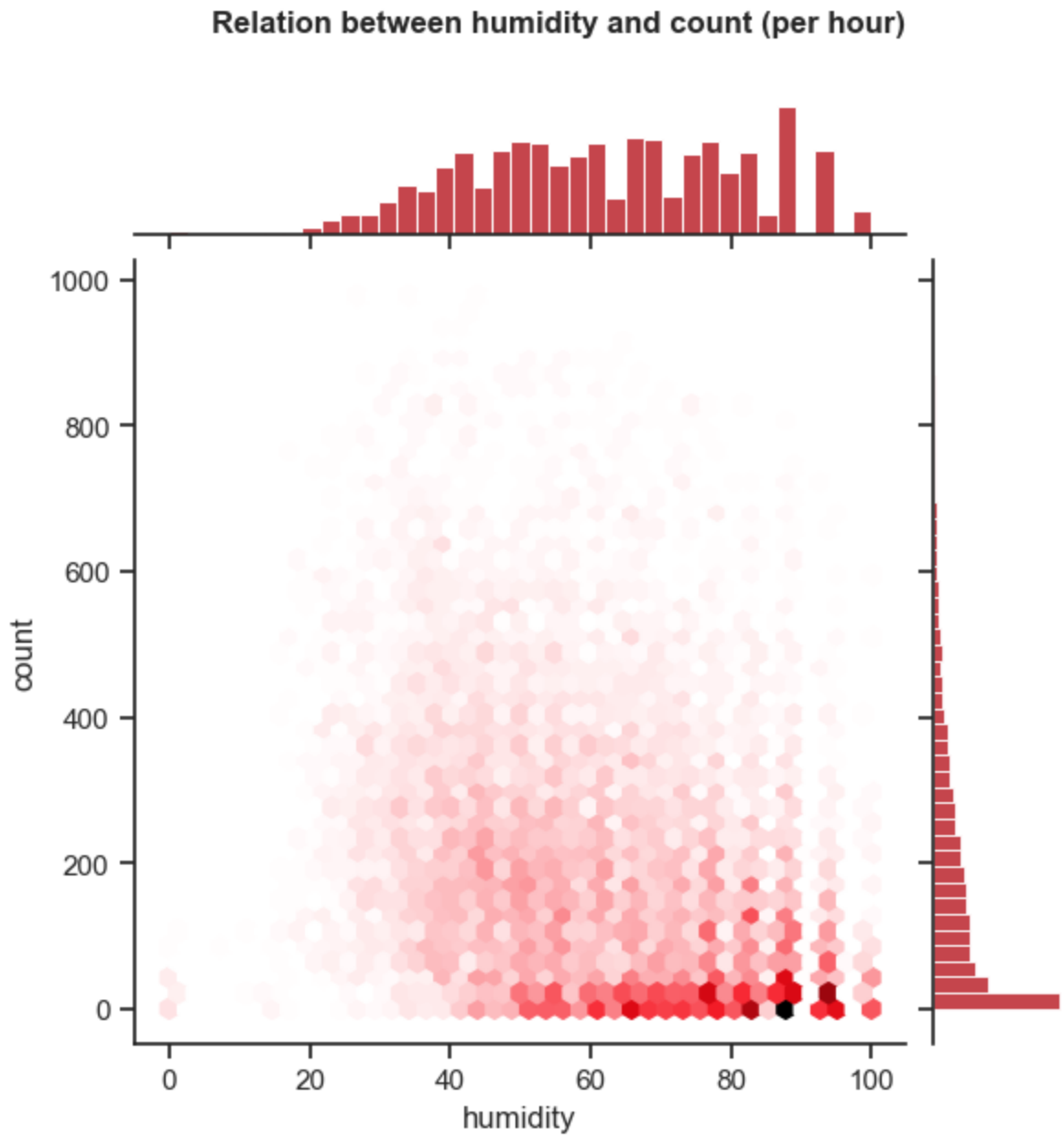
In [744…

```
plt.figure(figsize = (9,5))
sns.kdeplot(data=data, x="count", hue="year", fill=True, common_norm=False,  palet
plt.title("Kernel Density Estimations of cycle rent count per year")
plt.show()
```



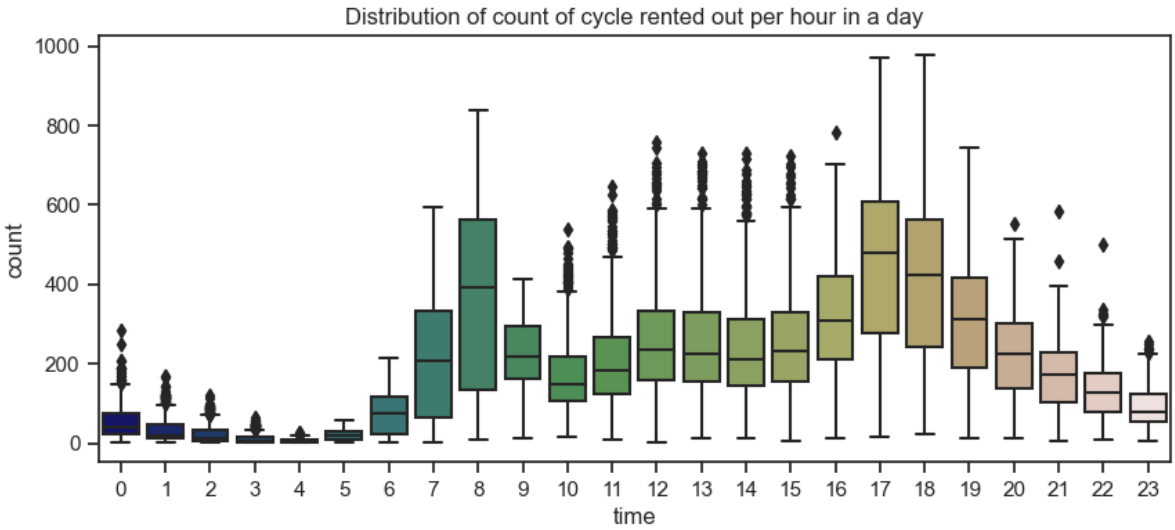
In [745…

```
sns.jointplot(x=data['humidity'], y=data['count'], kind="hex", color="#b20710")
plt.title("Relation between humidity and count (per hour)", loc = "right",pad= 90,
plt.show()
```



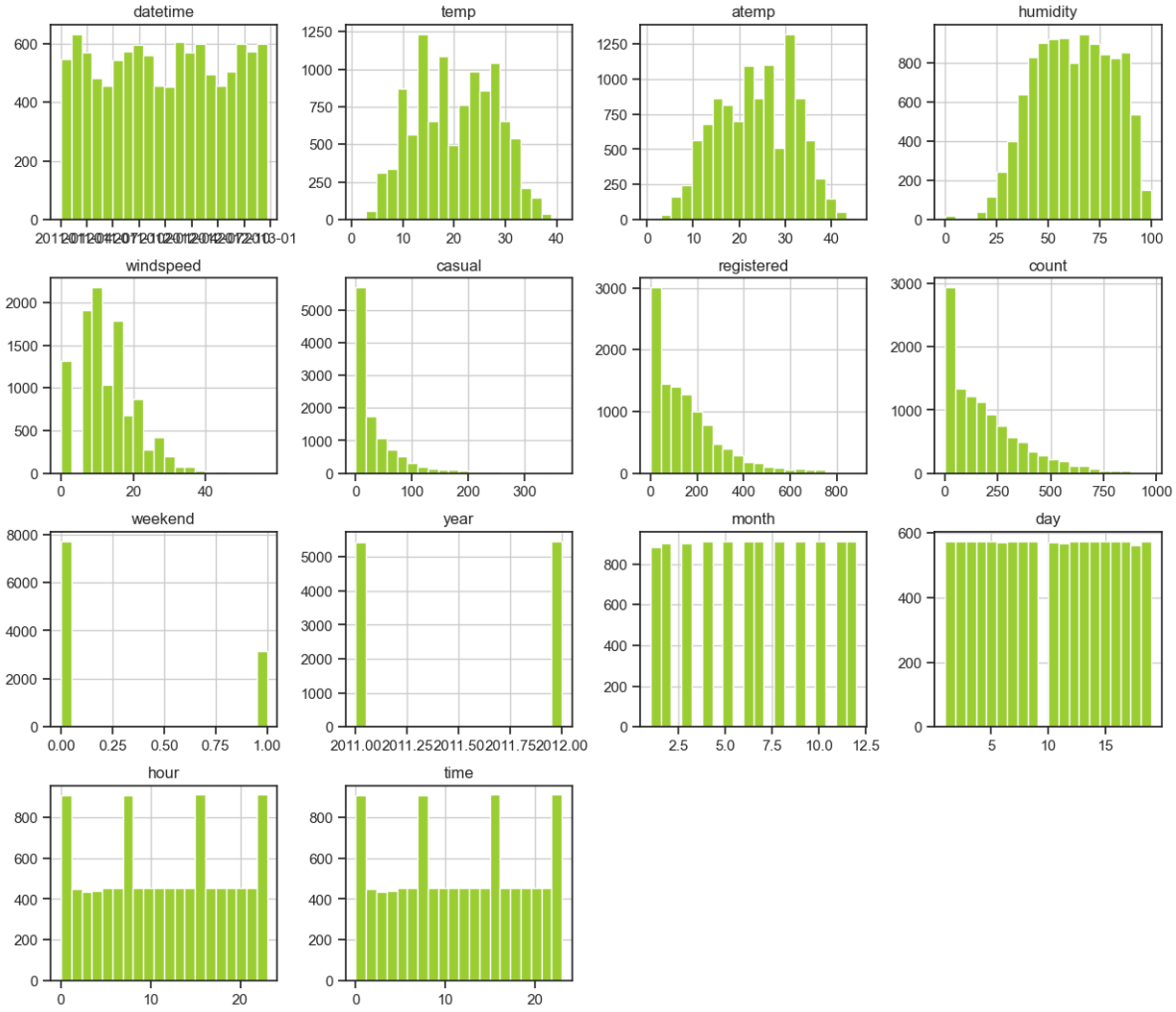
In [746...

```
data['time'] = data['time'].astype(str).apply(lambda x: x[:2]).astype(int)
plt.figure(figsize = (10,4))
sns.boxplot(data=data, x='time', y= 'count', palette='gist_earth')
plt.title("Distribution of count of cycle rented out per hour in a day")
plt.show()
```



Distribution of all numerical fields

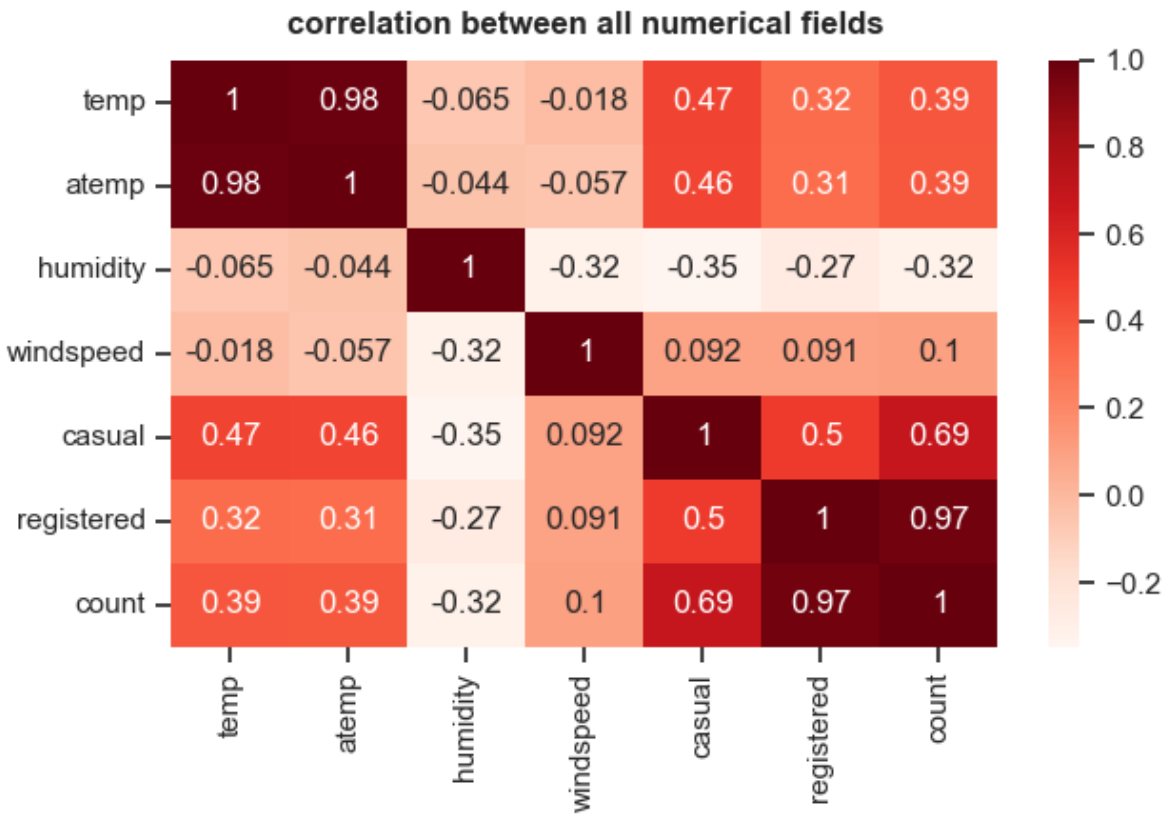
In [747…
```
data.hist(bins=20,figsize=(15,13),color= "yellowgreen")
plt.show()
```



- Whenever the humidity is less than 20, number of bikes rented is very very low.
- Whenever the temperature is less than 10, number of bikes rented is less.
- Whenever the windspeed is greater than 35, number of bikes rented is less.

correlation of all fields of dataset

In [748…
```
plt.figure(figsize = (7,4))
data_corr = data[["temp","atemp","humidity","windspeed","casual","registered","cou
sns.heatmap(data = data_corr,annot = True , cmap = "Reds")
plt.title("correlation between all numerical fields", loc = "center",pad= 10,fontw
plt.show(block = False)
```

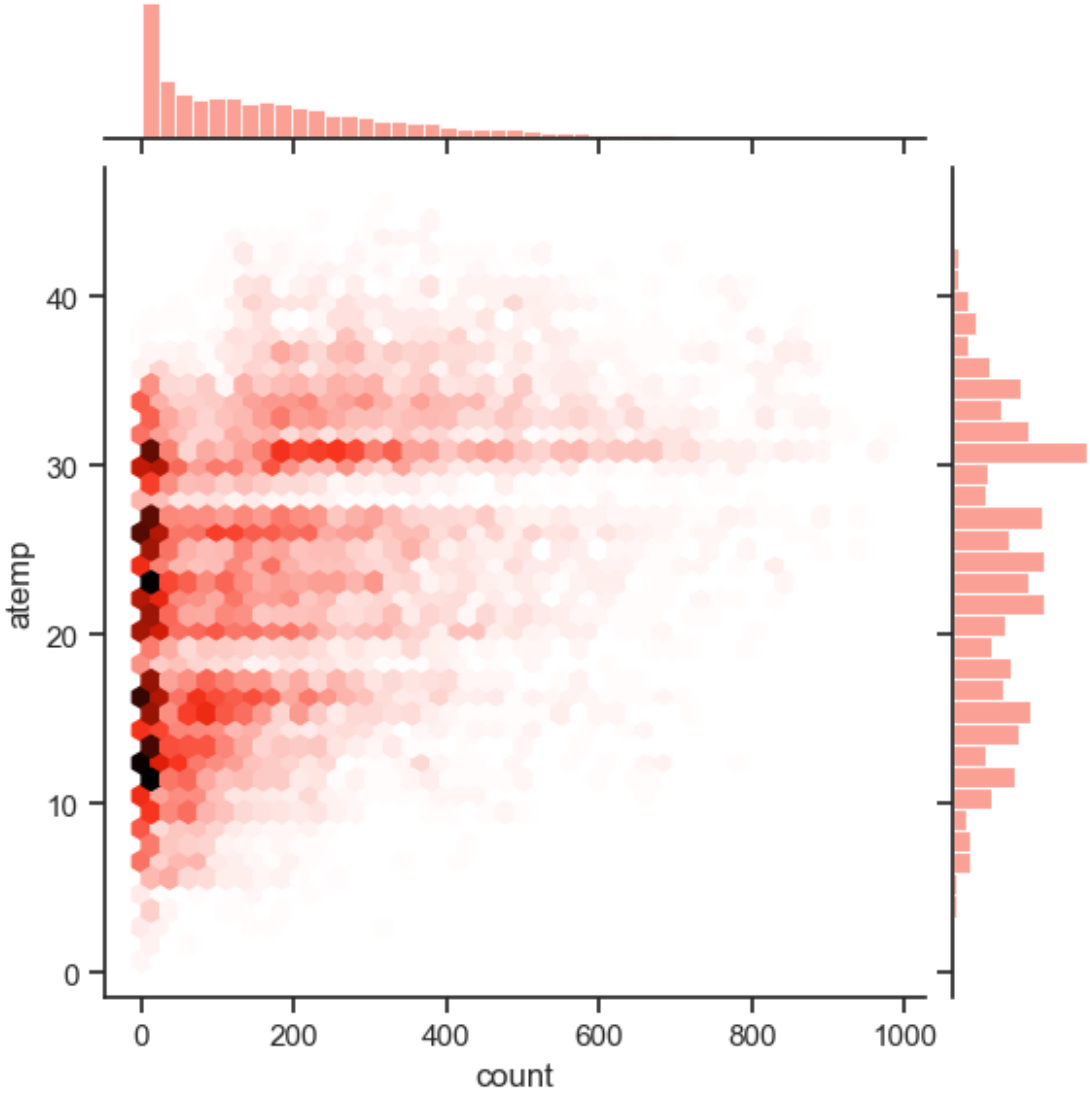


- Corrlation between Temperature and Number of Cycle Rented for all customers: 0.39
- Corrlation between Temperature and Number of Cycle Rented for casual subscribes : 0.46
- Corrlation between Temperature and Number of Cycle Rented for registered subscribes : 0.31
- Humidity has a negative correlation with number of cycles rented : 0.35

In [749…

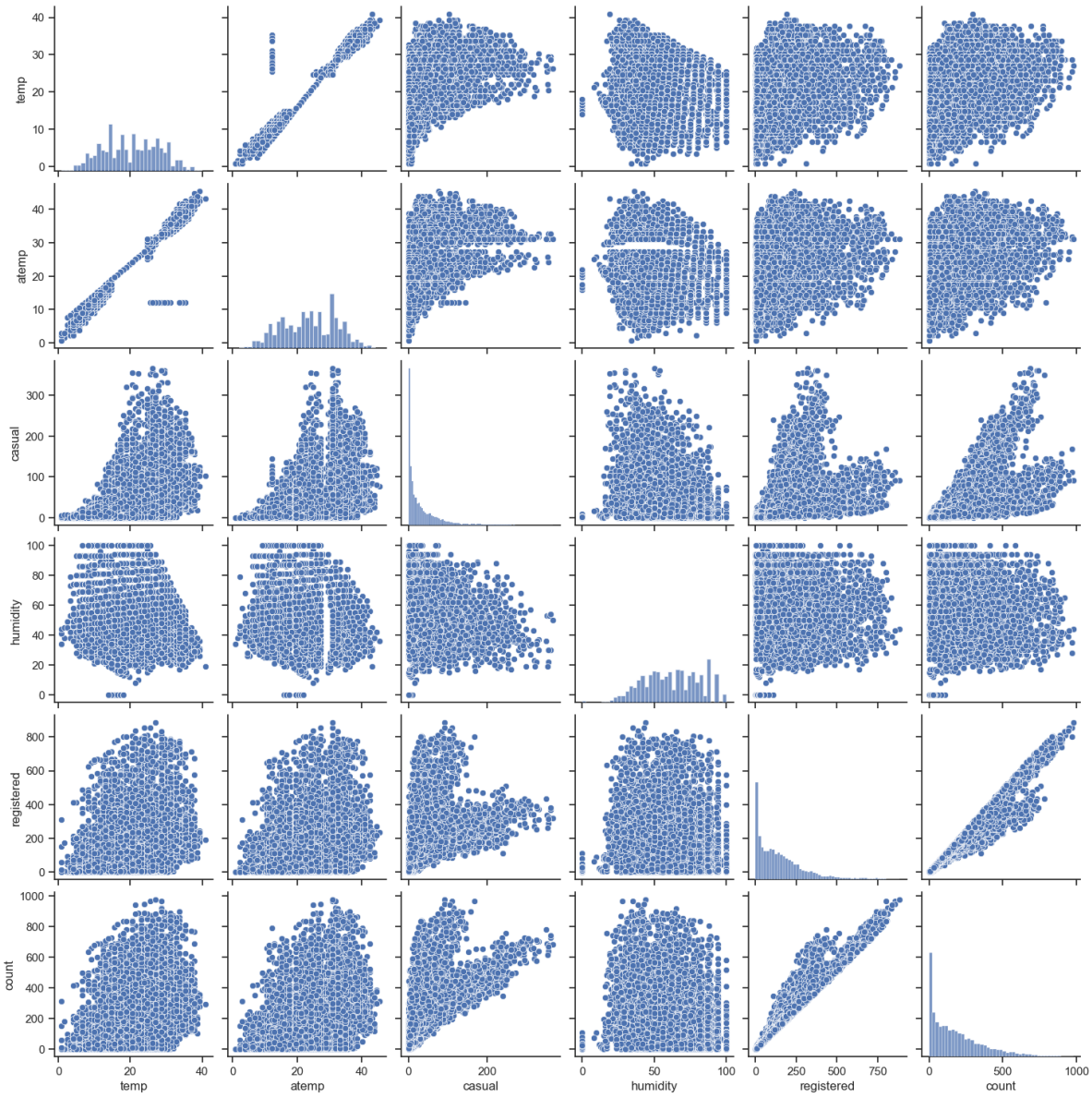
```
sns.jointplot(x=data['count'], y=data['atemp'], kind="hex", color="salmon")
plt.title("Relation between temperature felt and Total count of cycles rented per c
          loc = "center",pad= 90,fontweight="bold")
plt.show(block = False)
```

Relation between temperature felt and Total count of cycles rented per day



In [750…

```
sns.pairplot(data[["temp","atemp","casual","humidity","registered","count"]],palet
plt.show()
```



Bivariate Analysis: Observations

1. Temperature is the highest in Season 3 (when most cycles are rented).
2. Count of Rented cycles are lower at lower and higher temperatures and maximizes around the average temperature range 15 degrees and 27 degrees.
3. Count of rented Cycle are highest around humidity levels of 40 to 80. beyond those values, count of rented cycles drop.
4. The daily count of rented cycles have increased from year 2011 to 2012.
5. Cycles are rented less at temperatures lower than 10 degrees and rises with rise of temperature and again starts falling as temperature reaches 30 degrees.
6. Tuesdays and Fridays are the days when the most cycles are rented. on Sundays, the least number of cycles are rented.
7. Cycles rented count are significantly lower for the first 3 months of both years, starts increasing from months 4 to 10, and falls again at months 11 and 12.
8. In a day, most cycles are rented around 5pm and 6pm, whereas they are least rented around 3am and 4am.
9. Count of casual cycles rented falls during the weekdays and maximizes on weekends.
10. Count of registered cycles rented maximizes during weekdays and falls in number on weekends.

Hypothesis Testing

Approach

1. Set up a function to return result on the basis of the significance value(0.05).
2. 2- Sample T-Test to check if Working Day has an effect on the number of electric cycles rented.
3. ANNOVA to check if No. of cycles rented is similar or different in different 1. weather 2. season
4. Chi-square test to check if Weather is dependent on the season

Basic approach for all the tests

1. Step-1 : Set up Null and alternate Hypothesis
2. Step-2 : Checking for basic assumpitons for the hypothesis
    A. whether data is continuous.
    B. nature of sample data
    C. distribution check using QQ plot
    D. homogeneity of variance (i.e., the variability of the data in each group is similar): It will be checked by levene's test
    E. The distribution is approximately normal: validate by shapiro wilk test
    F. If necessary
3. Step-3 : Define Test statistics; Distribution of T under H0.
4. Step-4: Decide the kind of test:
    A. Perform Two tailed t-test
    B. ANNOVA test
    C. Chi-square test
5. Step-5: Compute the p-value and fix value of alpha.
6. Step-6: Compare p-value and alpha.** Based on p-value, accept or reject H0.
    A. p-val > alpha : Accept H0
    B. p-val < alpha : Reject H0

In [751…

```
#Setting up a function to return result on the basis of the significance value(0.05
def result(p_value):
  if p_value < 0.05:
    print("Reject Null Hypothesis")
  else:
    print("Failed to Reject Null Hypothesis")
```

2- Sample T-Test to check if Working Day has an effect on the number

of electric cycles rented

APPROACH

Step-1 : Hypothesis formulation

1. Null Hypothesis (H0) - number of electric cycles rented on working days and non working days are same.
2. Alternate Hypothesis (HA) - number of electric cycles rented on working days and non working days are different.

Step-2 : Checking for basic assumpitons for the hypothesis

1. Distribution check using QQ Plot
2. Homogeneity of Variances using Lavene's test
3. Chech normal distributions using shapiro wilk test

Step-3 : Define Test statistics; Distribution of T under H0.

1. Observe test statistic while performing a T-Test follows Tdistribution.

Step-4: Decide the kind of test.

1. Perform Two tailed t-test

Step-5: Compute the p-value and fix value of alpha.

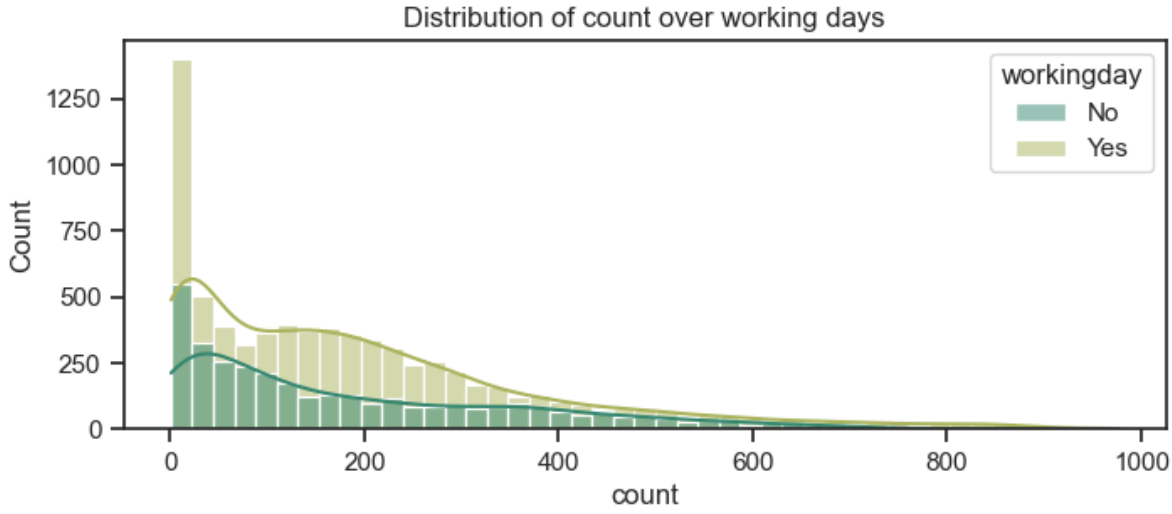
1. compute t-test value using the ttest function using scipy.stats. set *alpha to be 0.05*

Step-6: Compare p-value and alpha.

1. Based on p-value, accept or reject H0.
   A. p-val > alpha : Accept H0
   B. p-val < alpha : Reject H0

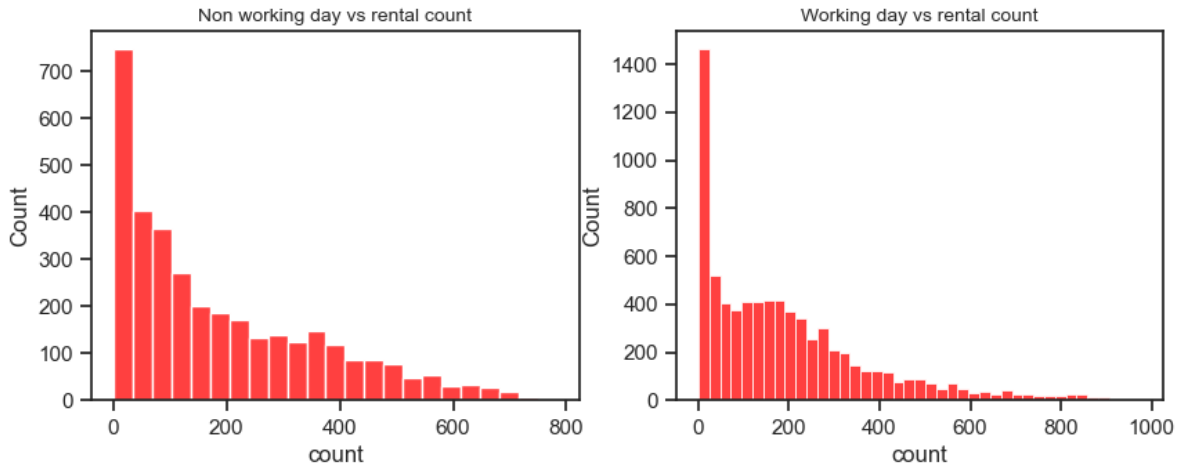
In [752...
```
data_non_working = data[data["workingday"]=="No"]["count"]
data_working = data.loc[data["workingday"]=="Yes","count"]
```

In [753...
```
#Visual analysis of count distribution over working and non-working days
plt.figure(figsize = (8,3))
sns.histplot(x=data["count"],kde=True,hue=data["workingday"],palette='gist_earth'
plt.title("Distribution of count over working days")
plt.show()
```



In [754...
```
# working or non-working days vs rental count
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(10,3))
```

```
fig.subplots_adjust(top=1)
sns.histplot(data= data_non_working,ax= axis[0],color = "red")
axis[0].set_title("Non working day vs rental count", pad=5, fontsize=10)
sns.histplot(data= data_working,ax= axis[1],color = "red")
axis[1].set_title("Working day vs rental count", pad=5, fontsize=10)
plt.show()
```



In [755…

```
# Checking Homogeneity of Variances using Lavene's test
# Null hypothesis(Ho) : Samples have similar variances
# Alternate Hypothesis(Ha) : samples have different variances
levene_stat, p_value =stats.levene(data_non_working ,data_working)
result(p_value)
```

```
Failed to Reject Null Hypothesis
```

In [756…

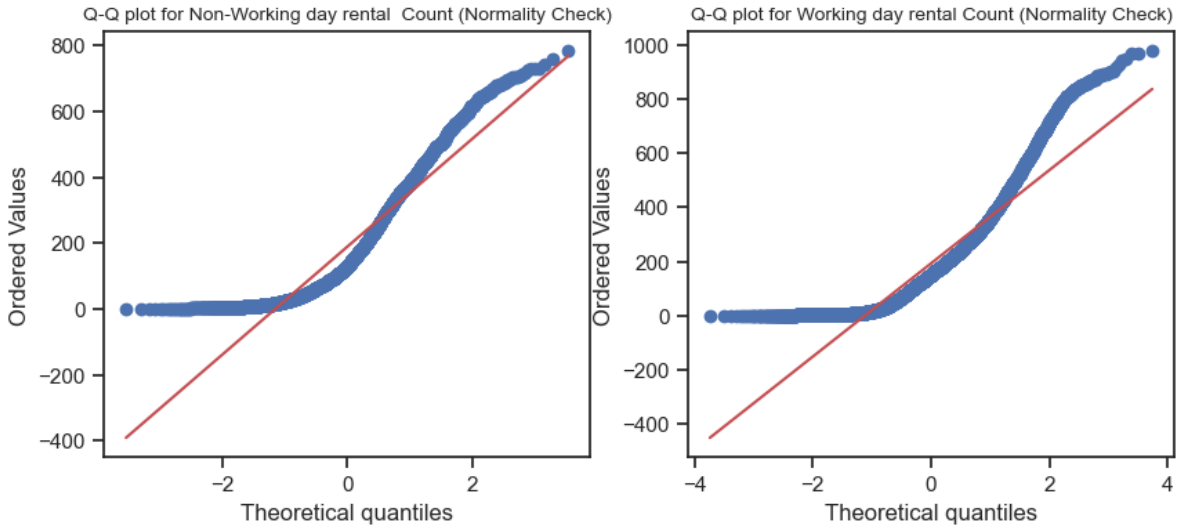
```
# checking distribution of normality using shapiro wilk test
# Null hypothesis(Ho) : Sample is from the normal distributions
# Alternate Hypothesis(Ha): Sample is not from the normal distributions.
shapiro_stat, p_value =stats.shapiro(data_non_working)
result(p_value)
shapiro_stat, p_value =stats.shapiro(data_working)
result(p_value)
```

```
Reject Null Hypothesis
Reject Null Hypothesis
```

As null hypothesis has been rejected, this means that data samples does not follow normal distribution. Now, verify the above result using Q-Q plots.

In [757…

```
#Construct QQ plots
plt.figure(figsize=(10,4))
plt.subplot(1,2,1)
stats.probplot(data_non_working, plot= plt, dist="norm")
plt.title('Q-Q plot for Non-Working day rental  Count (Normality Check)', fontsize
plt.subplot(1,2,2)
stats.probplot(data_working, plot= plt, dist="norm")
plt.title('Q-Q plot for Working day rental Count (Normality Check)', fontsize=10)
plt.show()
```



As observed from graphs, samples does not follow normal distribution, so taking into consideration that t-test is tolarent towards samples beings not normaly distributed, apply 2-sample t_stat test.

In [758…

```
# 2-sample t_stat test
t_stat, p_value = stats.ttest_ind(data_non_working,data_working)
print("p_value :",p_value )
result(p_value)
```

```
p_value : 0.22644804226361348
Failed to Reject Null Hypothesis
```

Conclusion:

number of electric cycles rented on working days and non working days are significantly same.
In other words, working and non working day alone don't stand any statistical significance on
the count of rentals.

ANNOVA to check if No. of cycles rented on weathers are similar or different

1: Clear, Few clouds, partly cloudy, partly cloudy

2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist

3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds

4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

APPROACH

Step-1 : Hypothesis formulation

1. Null Hypothesis (H0) - number of electric cycles rented on diferent weathers are same.

2. Alternate Hypothesis (HA) - number of electric cycles rented on diferent weathers are significantly different.

Step-2 : Checking for basic assumpitons for the hypothesis

1. The data are independent.Distribution check using QQ Plot
2. Homogeneity of Variances using Lavene's test
3. Check normal distributions using shapiro wilk test
4. Normality Distribution check using QQ Plot
5. check whether there is a statistical difference between the medians using Kruskal-Wallis test.
6. If the distribution is not normal, use BOX-COX transform to transform it to normal distribution.

Step-3 : Define Test statistics; Distribution of T under H0.

1. The test statistic for a One-Way ANOVA is denoted as F.
2. For an independent variable with k groups,the F statistic evaluates whether the group means are significantly different.

Step-4: Decide the kind of test.

1. Perform ANNOVA test

Step-5: Compute the p-value and fix value of alpha.

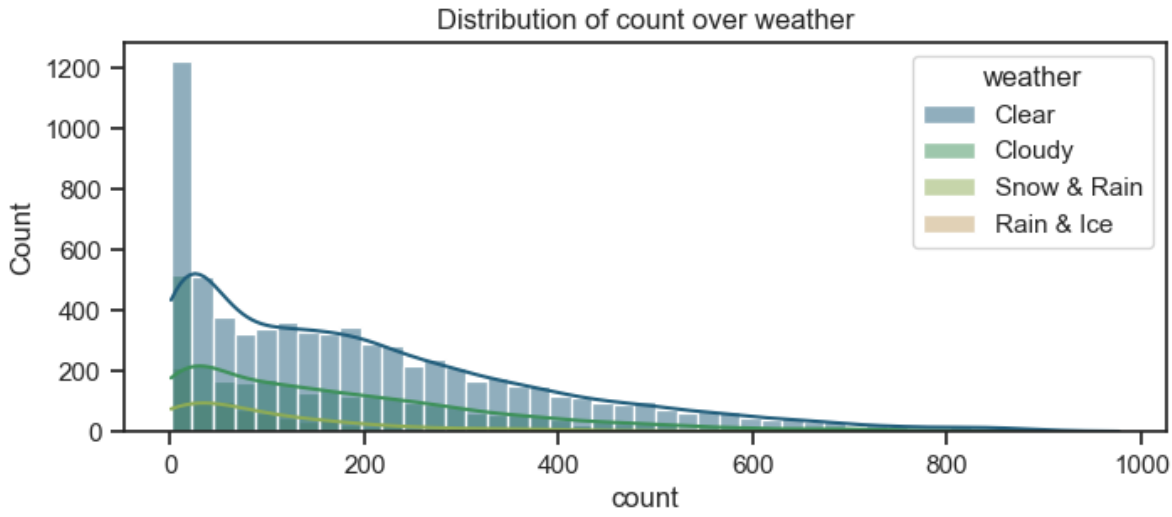
1. computing the anova-test p-value using the f_oneway function using scipy.stats. set *alpha to be 0.05*

Step-6: Compare p-value and alpha.

1. Based on p-value, accept or reject H0.
   A. p-val > alpha : Accept H0
   B. p-val < alpha : Reject H0

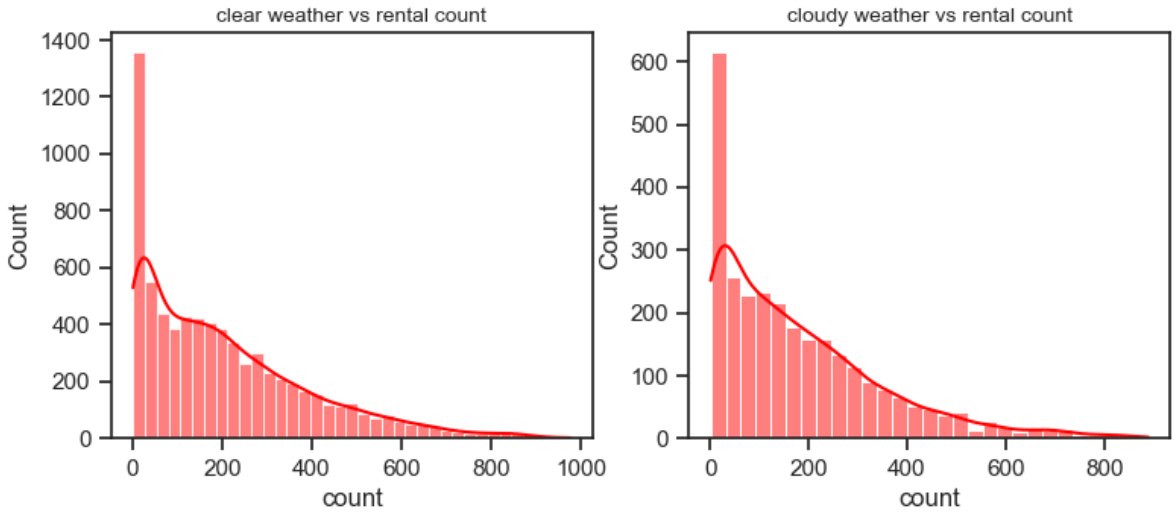
In [759…
```
weather_clear = data.loc[data["weather"]=="Clear","count"]
weather_cloudy = data.loc[data["weather"]=="Cloudy","count"]
weather_snow_rain = data.loc[data["weather"]=="Snow & Rain","count"]
weather_rain_ice = data.loc[data["weather"]=="Rain & Ice","count"]
```

In [760…
```
#Visual analysis
plt.figure(figsize = (8,3))
sns.histplot(x=data["count"],kde=True,hue=data["weather"],palette='gist_earth' )
plt.title("Distribution of count over weather")
plt.show()
```



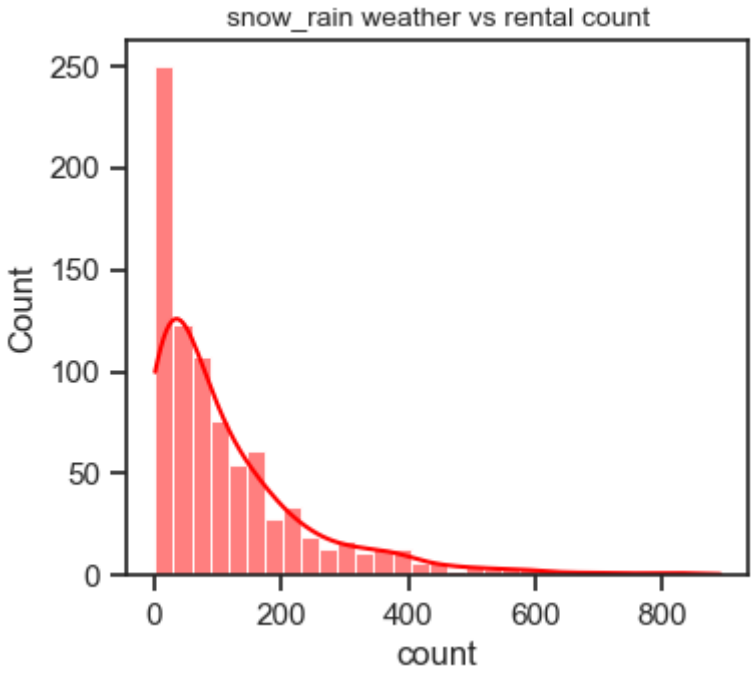
In [761…

```
# weathers vs rental counts
fig, axis = plt.subplots(ncols=2, figsize=(9,3))
fig.subplots_adjust(top=1)
sns.histplot(data= weather_clear,kde= True,ax= axis[0],color = "red")
axis[0].set_title("clear weather vs rental count", pad=5, fontsize=10)
sns.histplot(data= weather_cloudy,kde= True,ax= axis[1],color = "red")
axis[1].set_title("cloudy weather vs rental count", pad=5, fontsize=10)
plt.show()
```



In [762…

```
# weathers vs rental counts
fig, axis = plt.subplots(figsize=(4,3))
fig.subplots_adjust(top=1)
sns.histplot(data= weather_snow_rain,kde= True,color = "red")
plt.title("snow_rain weather vs rental count", pad=5, fontsize=10)
plt.show()
```



In [763…

```
# Checking Homogeneity of Variances using Lavene's test
# Null hypothesis(Ho) : Samples have similar variances
# Alternate Hypothesis(Ha) : samples have different variances
levene_stat, p_value =stats.levene(weather_clear,weather_cloudy,weather_snow_rain,
result(p_value)
```

```
Reject Null Hypothesis
```

In [764…

```
# checking distribution of normality using shapiro wilk test
# Null hypothesis(Ho) : Sample is from the normal distributions
# Alternate Hypothesis(Ha): Sample is not from the normal distributions.
shapiro_stat, p_value =stats.shapiro(weather_clear)
result(p_value)
shapiro_stat, p_value =stats.shapiro(weather_cloudy)
result(p_value)
shapiro_stat, p_value =stats.shapiro(weather_snow_rain)
result(p_value)
```

```
Reject Null Hypothesis
Reject Null Hypothesis
Reject Null Hypothesis
```

As rain and ice weather (weather 4) has only one data point. For shapiro test to be done, atleast 3 datapoints are required.

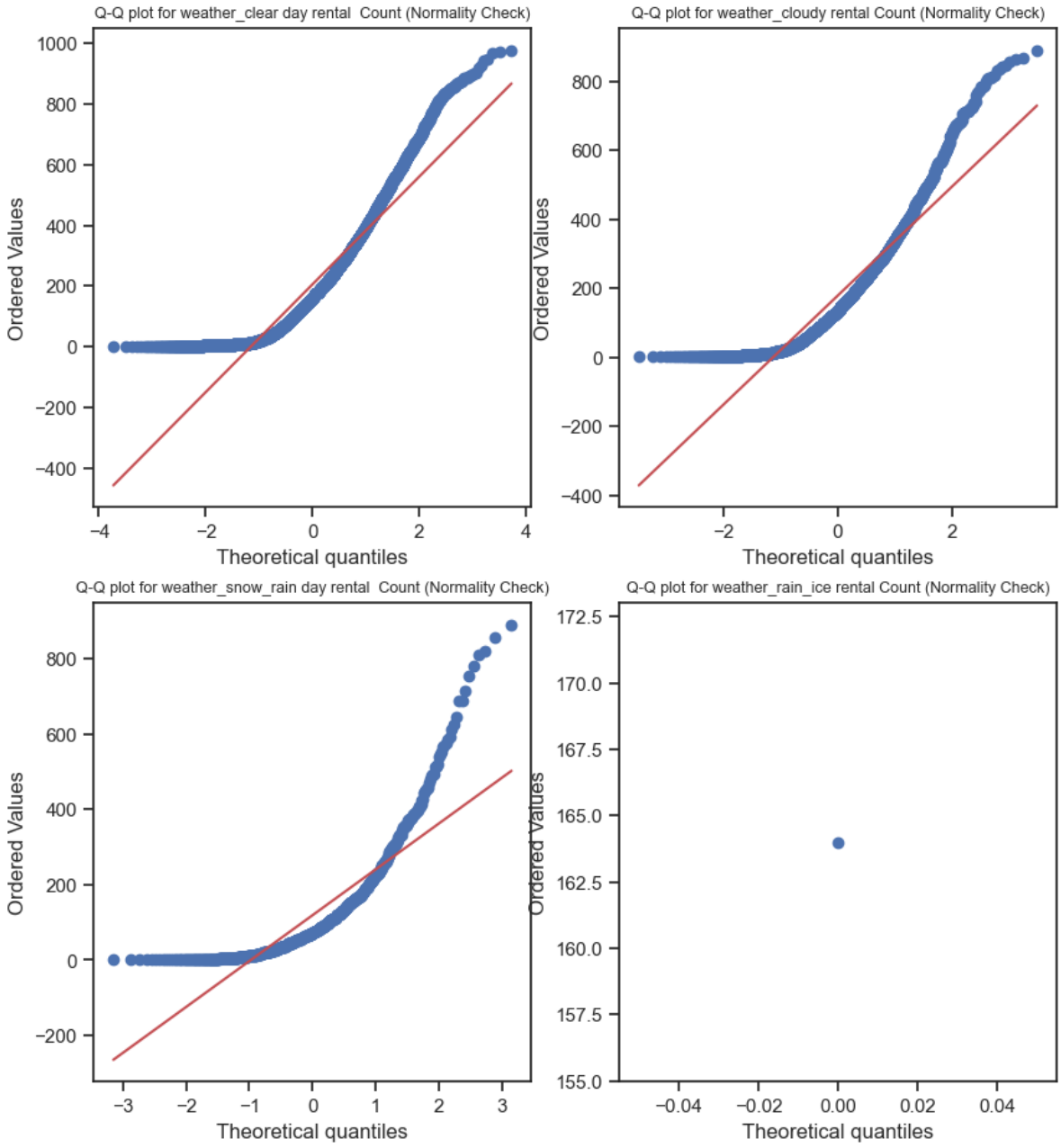
As null hypothesis has been rejected, this means that data samples does not follow normal distribution.

Now, verify the above result using Q-Q plots.

In [765…

```
# construct QQ plots
plt.figure(figsize=(10,11))
plt.subplot(2,2,1)
stats.probplot(weather_clear, plot= plt, dist="norm")
plt.title('Q-Q plot for weather_clear day rental  Count (Normality Check)', fontsi
plt.subplot(2,2,2)
stats.probplot(weather_cloudy, plot= plt, dist="norm")
plt.title('Q-Q plot for weather_cloudy rental Count (Normality Check)', fontsize=9
plt.subplot(2,2,3)
stats.probplot(weather_snow_rain, plot= plt, dist="norm")
```

```
plt.title('Q-Q plot for weather_snow_rain day rental  Count (Normality Check)', fo
plt.subplot(2,2,4)
stats.probplot(weather_rain_ice, plot= plt, dist="norm")
plt.title('Q-Q plot for weather_rain_ice rental Count (Normality Check)', fontsize
plt.show()
```



Kruskal-Wallis test

Kruskal-Wallis test is a non-parametric test and an alternative to One-Way ANNOVA.
By non-parametric we mean, the data is not assumed to become from a particular distribution.
The main objective of this test is used to determine whether there is a statistical difference
between the medians of at least three independent groups.

In [766…

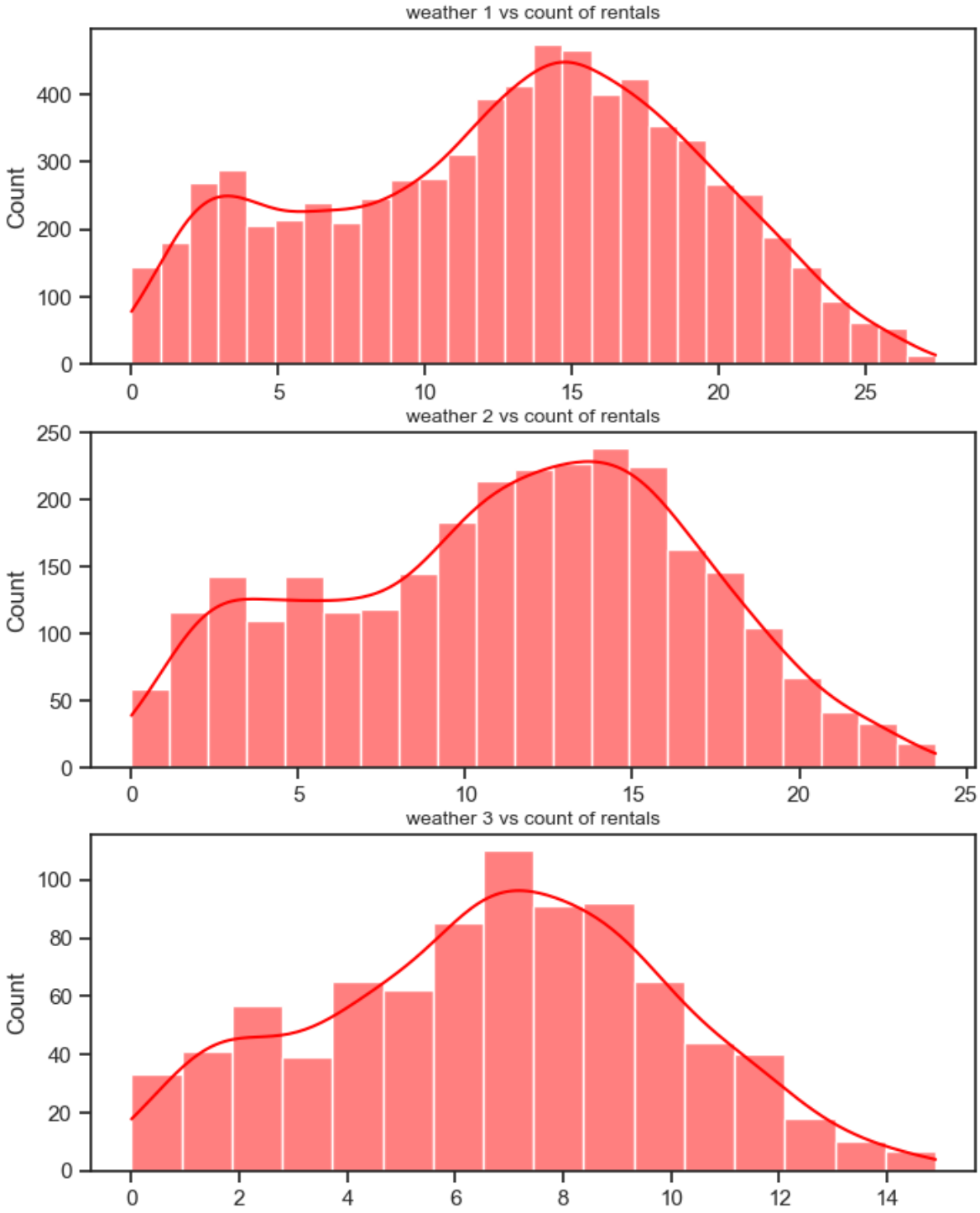
```
# check Kruskal-Wallis test
# The null hypothesis: The median is the same for all the data groups.
# The alternative hypothesis: The median is not equal for all the data groups.
kruskal_stat, p_value =stats.kruskal(weather_clear,weather_cloudy,weather_snow_rai
result(p_value)
```

Reject Null Hypothesis

As null hypothesis has been rejected, Kruskal-Wallies suggestes there is a effect of weather
in the count of the cycles rented.The varience of the samples are not similar.Also observed from
histogram and shapiro test the samples dont follow normal distribution. In order to make the the
sample set more compactable to ANNOVA text, BOX-COX Transform is applied.

In [767…

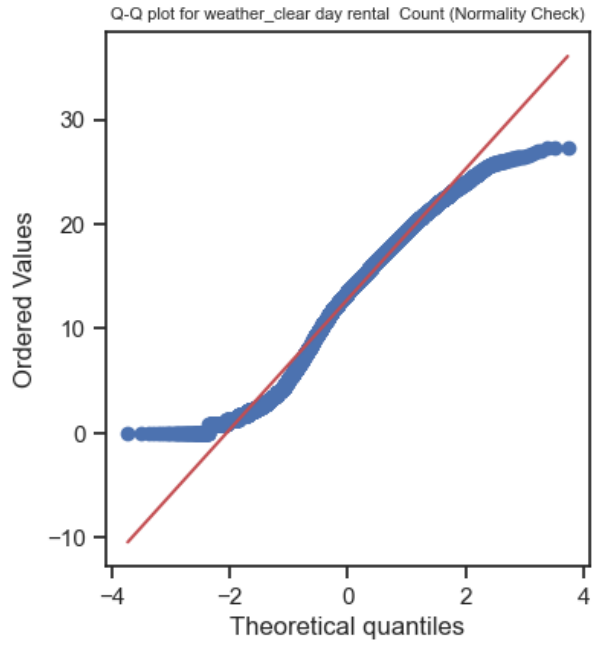
```
# BOX-COX Transform

weather_1= stats.boxcox(weather_clear)[0]
weather_2= stats.boxcox(weather_cloudy)[0]
weather_3= stats.boxcox(weather_snow_rain)[0]

fig, axis = plt.subplots(nrows=3, figsize=(8,9))
fig.subplots_adjust(top=1)
sns.histplot(data= weather_1,kde= True,ax= axis[0], color = "red")
axis[0].set_title("weather 1 vs count of rentals", pad=5, fontsize=10)
sns.histplot(data= weather_2,kde= True,ax= axis[1], color = "red")
axis[1].set_title("weather 2 vs count of rentals", pad=5, fontsize=10)
sns.histplot(data= weather_3,kde= True,ax= axis[2], color = "red")
axis[2].set_title("weather 3 vs count of rentals", pad=5, fontsize=10)
plt.show()
```

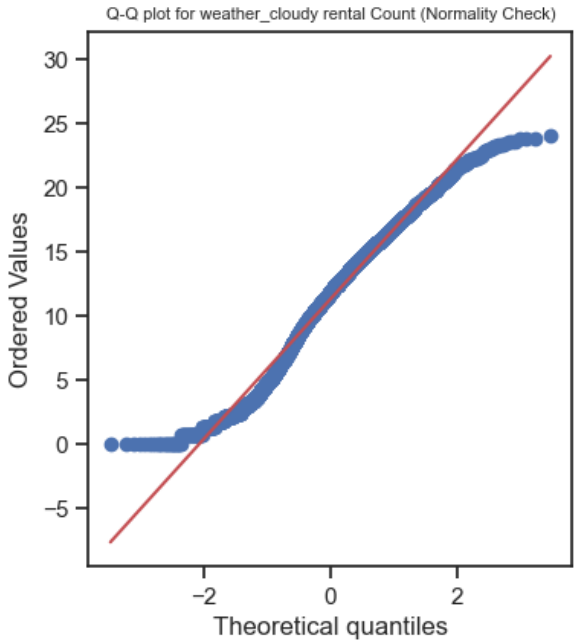


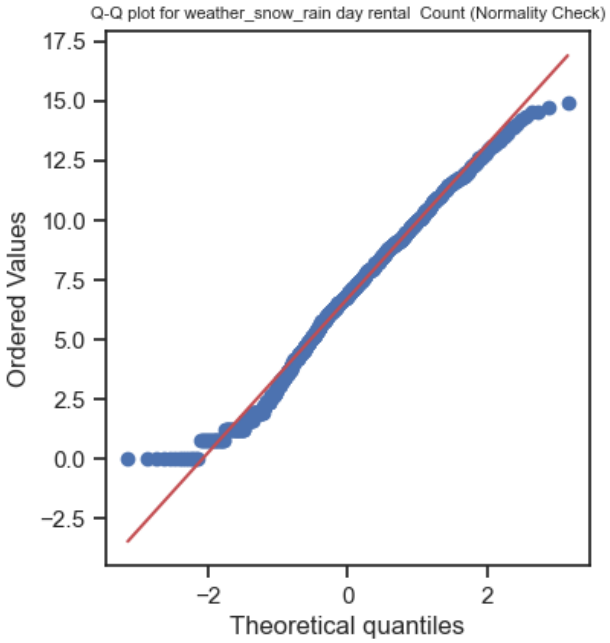
Now, the distribution is somewhat like normal distribution. Plot QQ plots again to verify

In [768…

```
# construct QQ plots
plt.figure(figsize=(9,10))
plt.subplot(2,2,1)
stats.probplot(weather_1, plot= plt, dist="norm")
plt.title('Q-Q plot for weather_clear day rental  Count (Normality Check)', fontsi
plt.subplot(2,2,2)
stats.probplot(weather_2, plot= plt, dist="norm")
plt.title('Q-Q plot for weather_cloudy rental Count (Normality Check)', fontsize=8
plt.subplot(2,2,3)
stats.probplot(weather_3, plot= plt, dist="norm")
plt.title('Q-Q plot for weather_snow_rain day rental  Count (Normality Check)', fo
plt.show()
```







If compare the above two QQ plots, it can be observed that first QQ plot does not follow normal distribution,however, after BOXCOX transform, QQ plots are following somewhat normal distribution.

Though the end datapoints are not meeting the normal distribution, majority of the datapoints have aligned with normal distribution. It seems safe to apply ANNOVA test on this sample.

In [769…

```
# apply ANNOVA test
f_stat, p_value = stats.f_oneway(weather_1,weather_2,weather_3)
print("p_value :",p_value )
result(p_value)
```

```
p_value : 3.4867243611236345e-181
Reject Null Hypothesis
```

```
***conclusion:***

number of electric cycles rented on different weather are
different.
In other words, weather have statistical significance on the count
of rentals.
```

ANNOVA to check if No. of cycles rented on seasons are similar or different

1: spring

2: summer

3: fall

4: winter

APPROACH

Step-1 : Hypothesis formulation

1. Null Hypothesis (H0) - number of electric cycles rented on diferent seasons are same.
2. Alternate Hypothesis (HA) - number of electric cycles rented on diferent seasons are significantly different.

Step-2 : Checking for basic assumpitons for the hypothesis

1. The data are independent.Distribution check using QQ Plot
2. Homogeneity of Variances using Lavene's test
3. Check normal distributions using shapiro wilk test
4. Normality Distribution check using QQ Plot
5. check whether there is a statistical difference between the medians using Kruskal-Wallis test.
6. If the distribution is not normal, use BOX-COX transform to transform it to normal distribution.

Step-3 : Define Test statistics; Distribution of T under H0.

1. The test statistic for a One-Way ANOVA is denoted as F.
2. For an independent variable with k groups,the F statistic evaluates whether the group means are significantly different.

Step-4: Decide the kind of test.

1. Perform ANNOVA test

Step-5: Compute the p-value and fix value of alpha.

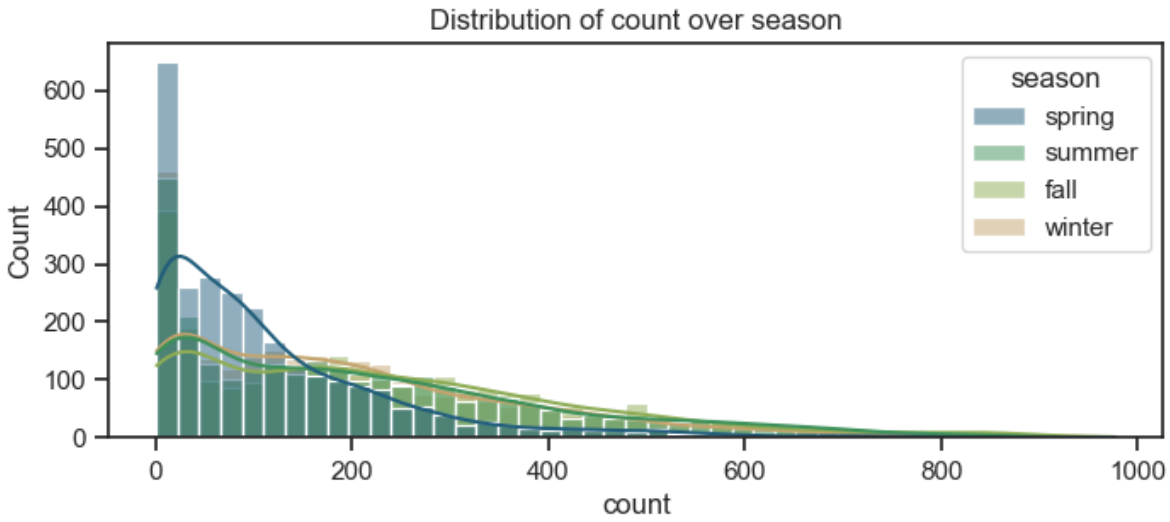
1. computing the anova-test p-value using the f_oneway function using scipy.stats. set *alpha to be 0.05*

Step-6: Compare p-value and alpha.

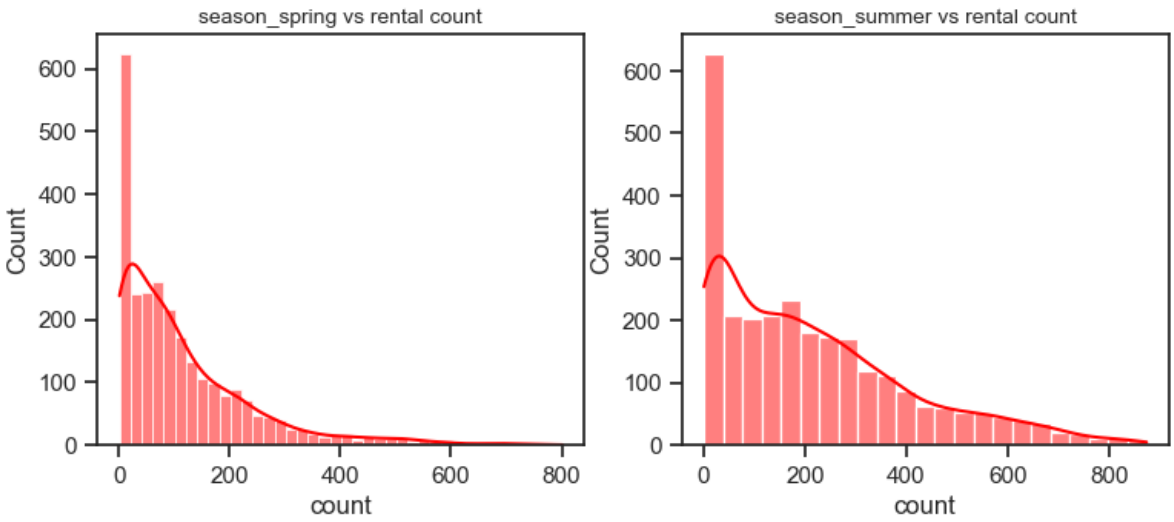
1. Based on p-value, accept or reject H0.
   A. p-val > alpha : Accept H0
   B. p-val < alpha : Reject H0

In [770…
```
season_spring = data.loc[data["season"]=="spring","count"]
season_summer = data.loc[data["season"]=="summer","count"]
season_fall = data.loc[data["season"]=="fall","count"]
season_winter = data.loc[data["season"]=="winter","count"]
```

In [771…
```
#Visual analysis
plt.figure(figsize = (8,3))
sns.histplot(x=data["count"],kde=True,hue=data["season"],palette='gist_earth' )
plt.title("Distribution of count over season")
plt.show()
```

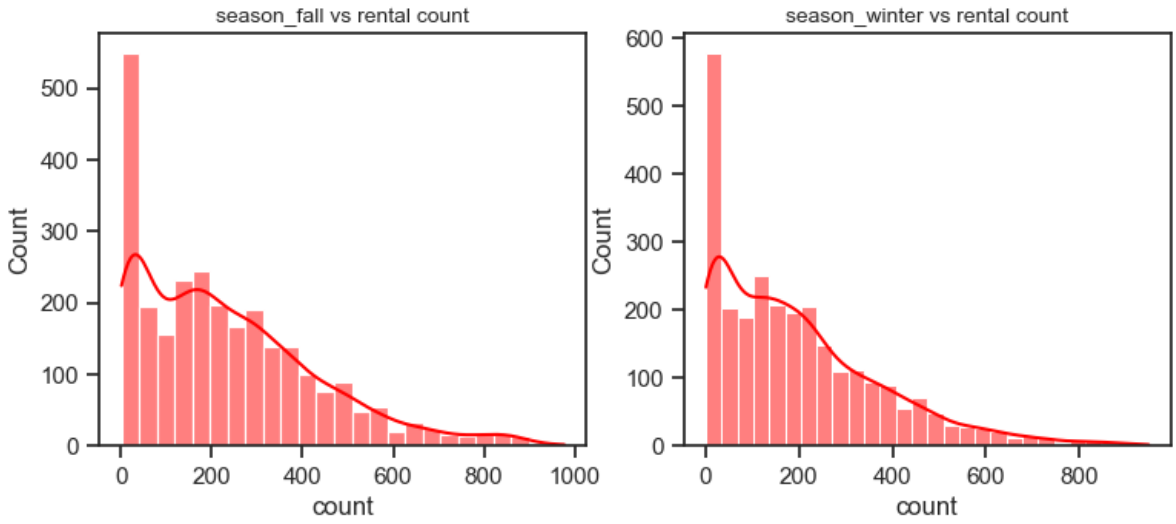


In [772…
```
# seasons vs rental counts
fig, axis = plt.subplots(ncols=2, figsize=(9,3))
fig.subplots_adjust(top=1)
sns.histplot(data= season_spring,kde= True,ax= axis[0],color = "red")
axis[0].set_title("season_spring vs rental count", pad=5, fontsize=10)
sns.histplot(data= season_summer,kde= True,ax= axis[1],color = "red")
axis[1].set_title("season_summer vs rental count", pad=5, fontsize=10)
plt.show()
```



In [773…

```
# seasons vs rental counts
fig, axis = plt.subplots(ncols=2, figsize=(9,3))
fig.subplots_adjust(top=1)
sns.histplot(data= season_fall,kde= True,ax= axis[0],color = "red")
axis[0].set_title("season_fall vs rental count", pad=5, fontsize=10)
sns.histplot(data= season_winter,kde= True,ax= axis[1],color = "red")
axis[1].set_title("season_winter vs rental count", pad=5, fontsize=10)
plt.show()
```



In [774…

```
# Checking Homogeneity of Variances using Lavene's test
# Null hypothesis(Ho) : Samples have similar variances
# Alternate Hypothesis(Ha) : samples have different variances
levene_stat, p_value =stats.levene(season_spring,season_summer,season_fall,season_
result(p_value)
```

```
Reject Null Hypothesis
```

In [775…

```
# checking distribution of normality using shapiro wilk test
# Null hypothesis(Ho) : Sample is from the normal distributions
# Alternate Hypothesis(Ha): Sample is not from the normal distributions.
shapiro_stat, p_value =stats.shapiro(season_spring)
result(p_value)
shapiro_stat, p_value =stats.shapiro(season_summer)
result(p_value)
shapiro_stat, p_value =stats.shapiro(season_fall)
result(p_value)
shapiro_stat, p_value =stats.shapiro(season_winter)
result(p_value)
```

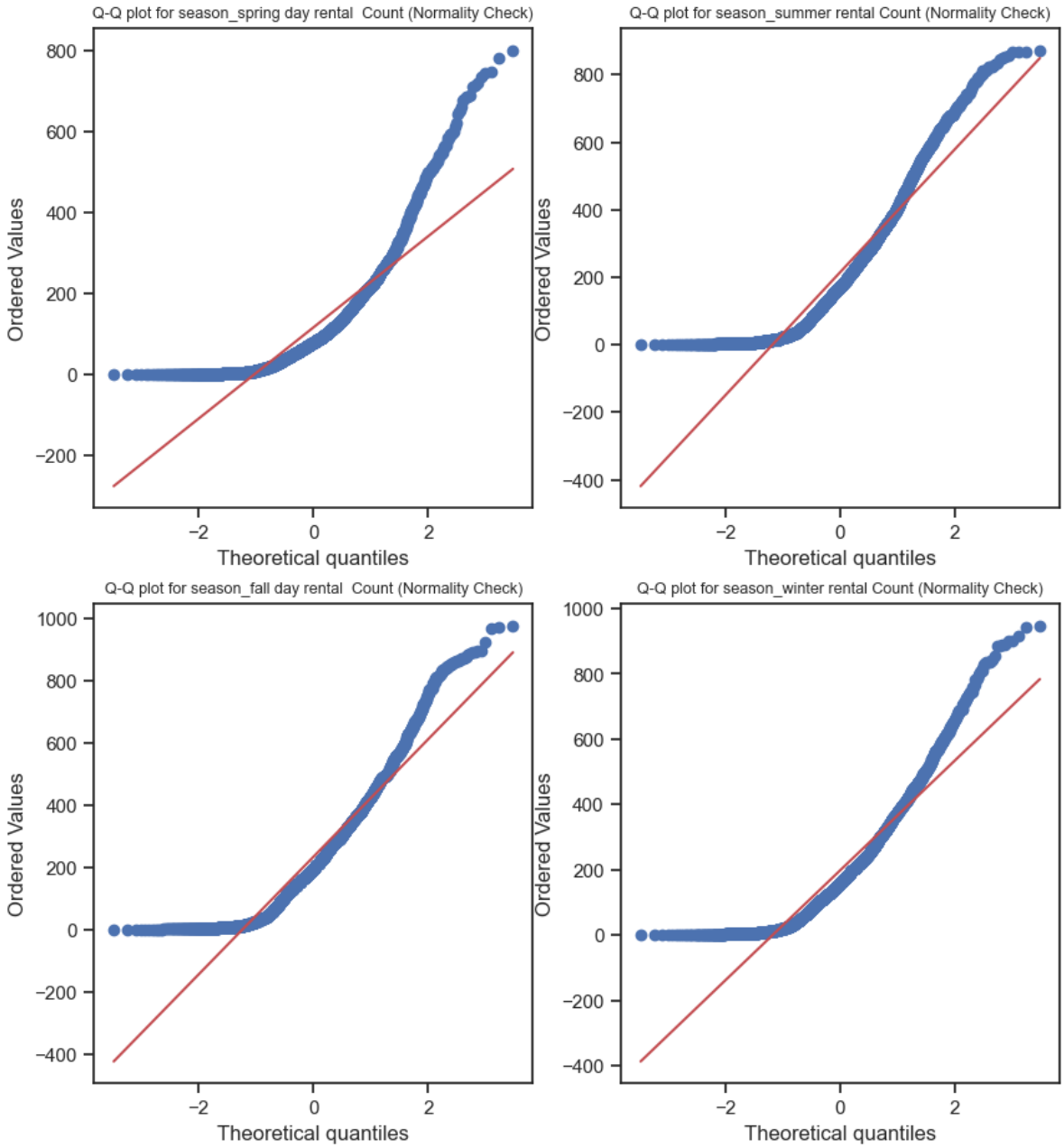
```
Reject Null Hypothesis
Reject Null Hypothesis
Reject Null Hypothesis
Reject Null Hypothesis
```

As null hypothesis has been rejected, this means that data samples does not follow normal
distribution.
Now, verify the above result using Q-Q plots.

In [776…

```
# construct QQ plots
plt.figure(figsize=(10,11))
plt.subplot(2,2,1)
stats.probplot(season_spring, plot= plt, dist="norm")
plt.title('Q-Q plot for season_spring day rental  Count (Normality Check)', fontsi
```

```
plt.subplot(2,2,2)
stats.probplot(season_summer, plot= plt, dist="norm")
plt.title('Q-Q plot for season_summer rental Count (Normality Check)', fontsize=9)
plt.subplot(2,2,3)
stats.probplot(season_fall, plot= plt, dist="norm")
plt.title('Q-Q plot for season_fall day rental  Count (Normality Check)', fontsize
plt.subplot(2,2,4)
stats.probplot(season_winter, plot= plt, dist="norm")
plt.title('Q-Q plot for season_winter rental Count (Normality Check)', fontsize=9)
plt.show()
```



QQ plots showing that data points are not following normal distribution.
Going ahead and testing Kruskal-Wallis test

In [777…

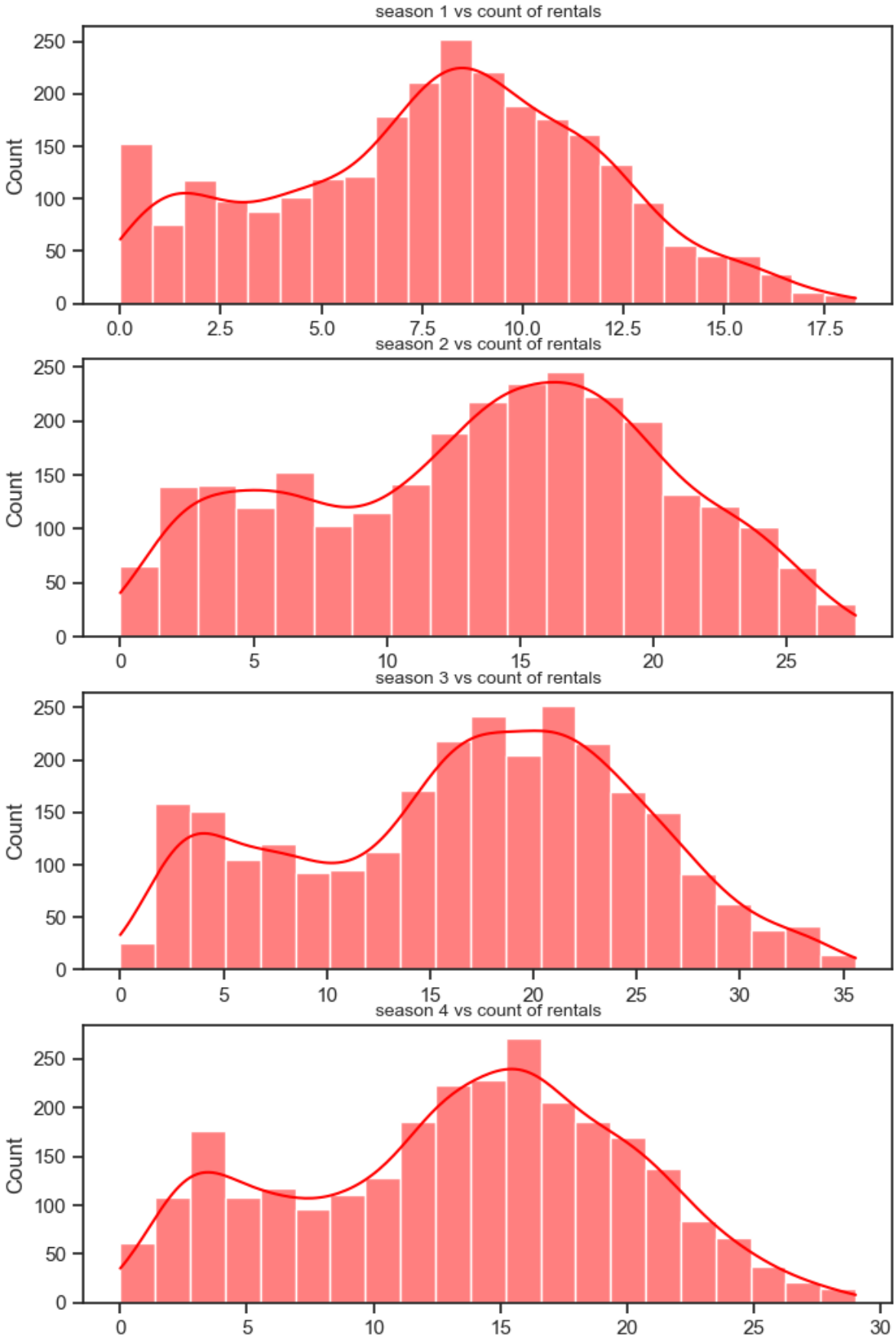
```
# check Kruskal-Wallis test
# The null hypothesis: The median is the same for all the data groups.
# The alternative hypothesis: The median is not equal for all the data groups.
kruskal_stat, p_value =stats.kruskal(season_spring,season_summer,season_fall,seasor
result(p_value)
```

Reject Null Hypothesis

As null hypothesis has been rejected, Kruskal-Wallies suggestes there is a effect of weather
in the count of the cycles rented.The varience of the samples are not similar.Also observed from
histogram and shapiro test the samples dont follow normal distribution. In order to make the the
sample set more compactable to ANNOVA text, BOX-COX Transform is applied.

In [778…

```
# BOX-COX Transform

season_1= stats.boxcox(season_spring)[0]
season_2= stats.boxcox(season_summer)[0]
season_3= stats.boxcox(season_fall)[0]
season_4= stats.boxcox(season_winter)[0]

fig, axis = plt.subplots(nrows=4, figsize=(8,11))
fig.subplots_adjust(top=1)
sns.histplot(data= season_1,kde= True,ax= axis[0], color = "red")
axis[0].set_title("season 1 vs count of rentals", pad=5, fontsize=10)
sns.histplot(data= season_2,kde= True,ax= axis[1], color = "red")
axis[1].set_title("season 2 vs count of rentals", pad=5, fontsize=10)
sns.histplot(data= season_3,kde= True,ax= axis[2], color = "red")
axis[2].set_title("season 3 vs count of rentals", pad=5, fontsize=10)
sns.histplot(data= season_4,kde= True,ax= axis[3], color = "red")
axis[3].set_title("season 4 vs count of rentals", pad=5, fontsize=10)
plt.show()
```

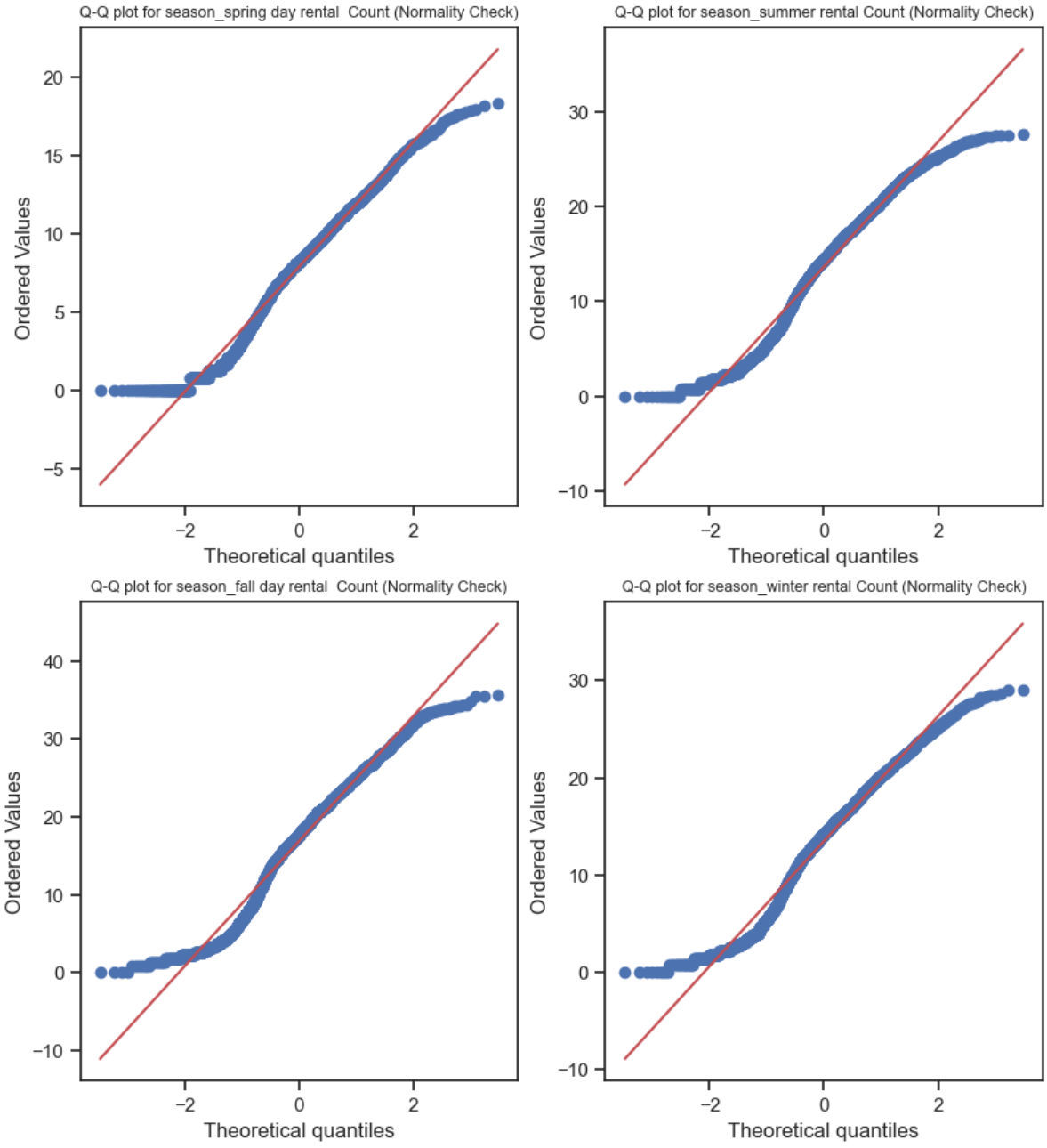


Now, the distribution is somewhat like normal distribution. Plot QQ plots again to verify

In [779…

```
# construct QQ plots
plt.figure(figsize=(10,11))
plt.subplot(2,2,1)
```

```
stats.probplot(season_1, plot= plt, dist="norm")
plt.title('Q-Q plot for season_spring day rental  Count (Normality Check)', fontsi
plt.subplot(2,2,2)
stats.probplot(season_2, plot= plt, dist="norm")
plt.title('Q-Q plot for season_summer rental Count (Normality Check)', fontsize=9)
plt.subplot(2,2,3)
stats.probplot(season_3, plot= plt, dist="norm")
plt.title('Q-Q plot for season_fall day rental  Count (Normality Check)', fontsize
plt.subplot(2,2,4)
stats.probplot(season_4, plot= plt, dist="norm")
plt.title('Q-Q plot for season_winter rental Count (Normality Check)', fontsize=9)
plt.show()
```



If compare the above two QQ plots, it can be observed that first QQ plot does not follow normal distribution,however, after BOXCOX transform, QQ plots are following somewhat normal distribution.

Though the end datapoints are not meeting the normal distribution, majority of the datapoints have

aligned with normal distribution. It seems safe to apply ANNOVA test on this sample.

In [780…
```
# apply ANNOVA test
f_stat, p_value = stats.f_oneway(season_1,season_2,season_3,season_4)
print("p_value :",p_value )
result(p_value)
```

```
p_value : 0.0
Reject Null Hypothesis
```

conclusion:

number of electric cycles rented on different seasons are different.
In other words, seasons have statistical significance on the count of rentals.

Weather is dependent on season or not?

(check between 2 predictor variable) : Chi-square test

The Chi-square statistic is a non-parametric (distribution free) tool designed to analyze group
differences when the dependent variable is measured at a nominal level. Like all non-parametric
statistics, the Chi-square is robust with respect to the distribution of the data. Specifically,
it does not require equality of variances among the study groups or homoscedasticity in the data.

APPROACH

STEP-1 : Set up Null Hypothesis

1. Null Hypothesis(Ho): The weather and seosons are not related in the population, the propotion of different weathers are same for all seasons
2. Alternate Hypothesis(Ha): The weather and seasons are related in the population. The propotion of different weathers are different for diferent seasons

STEP-2: Checking for basic assumpitons for the hypothesis (Non-Parametric Test)

1. Both variables are categorical.
2. All observations are independent and mutually exclusive.
3. Expected value of cells should be 5 or greater in at least 80% of cells

*As weather type 4 have only one entry and it doesnt satisfy assumption of chi- squared test. Hence, removing that entry from analysis

STEP-3: Define Test statistics; Distribution of T under H0.

1. The test statistic for a Chi- square test is denoted as T.
2. Under H0, the test statistic should follow Chi-Square Distribution.

STEP-4: Decide the kind of test.

1. performing chi-square test

STEP-5: Compute the p-value and fix value of alpha.

1. compute chi square-test p-value using the chi 2 function using scipy.stats. We set our alpha to be 0.05

STEP-6: Compare p-value and alpha.

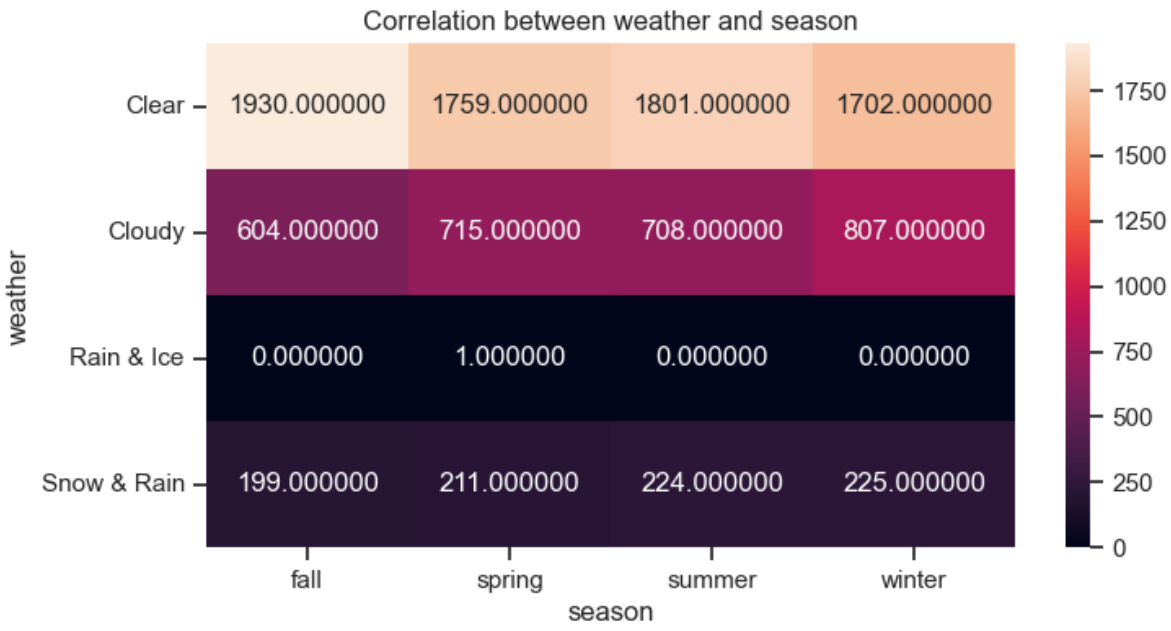
1. Based on p-value, accept or reject H0.
   A. p-val > alpha : Accept H0
   B. p-val < alpha : Reject H0

In [781…
```
weather_season = pd.crosstab(data['weather'],data['season'])
weather_season
```

Out[781]:

season	fall	spring	summer	winter
weather				
Clear	1930	1759	1801	1702
Cloudy	604	715	708	807
Rain & Ice	0	1	0	0
Snow & Rain	199	211	224	225

In [782…
```
plt.figure(figsize=(8,4))
sns.heatmap(data = weather_season, annot=True, fmt='f')
plt.title('Correlation between weather and season')
plt.show()
```



In [783…
```
data_weather = data.loc[data["weather"]!= "Rain & Ice","weather"]
data_weather_season = pd.crosstab(data_weather,data['season'])
data_weather_season
```

Out[783]:

season	fall	spring	summer	winter
weather				
Clear	1930	1759	1801	1702
Cloudy	604	715	708	807
Snow & Rain	199	211	224	225

In [784…
```
# Apply Chi-square test
chi_stat,p_value,df,exp_value = stats.chi2_contingency(data_weather_season)
print("p_value :",p_value )
result(p_value)
```

```
p_value : 2.8260014509929403e-08
Reject Null Hypothesis
```

```
***Conclusion:**

As null hypothesis has been rejected, Weather and seasons are co-
related.
The propotion of different weathers are different for diferent
seasons.
```

Insights

- Season - Count of rented bikes is maximum in fall season and minimum in spring season
- Weather - Count of rented bikes are maximum in weather 1 and minimum in weather 4.
- Temperature - Count of rented bikes is very high around moderate temperatures and lowers with fall in temperature.
- On working days, count of registered cycles rented is higher than that of casual cycles.
- On weekends, count of casual Cycles rented is higher than that of registered cycles.
- Humidity, windspeed, temperature and weather are correlated with season and impacts the count of cycles rented. Humidity has negative coorelation with cycle rent count.
- By using 2-sample T-Test we conclude that number of electric cycles rented on working days and non working days are significantly same. This means that total count of rentals is independent of working or non working day and they do not have any statistical significance.
- With ANNOVA we conclude season and weather have statistical significance on the count of rentals
- With chi squared test we conclude Weather and seasons are co-related. The proportion of different weathers are different for different seasons.

Recommendations

1. EDA suggests that cycle rented is very low during season 1. Yulu can rollout some challenges during this season, like New year goals on fitness and roll out exciting prizes for people participating in this season.
2. The count of cycle is very low during the night hours. Yulu can introduce some safety features in its app and also some promotional discounts for people renting yulu bikes from 12am - 6am, which can motivate people to ride Yulu bikes during this time frame.
3. The EDA proves that Temperature, humidity, windspeed and weather - all these environmental factors affect the count of bikes rented on Yulu. Even though Yulu can not change the weather conditions, Yulu can offer reduced pricing for people renting in these weather conditions (weather 2 & 3).
4. The count of bikes rented on holidays are significantly lower than on non-holidays. Yulu can set up bike centers near entertainment hotspots, like malls and amusemennt parks to hike up its rental count on weekends and holidays.
5. Yulu can introduce some early biker scheme (4am -7am), that would target the people who are motivated in physical fitness and also roll out some exciting prizes for people covering the longest distance rides.
6. Yulu can introduce exclusive benifits for registered users to get more users to register, which can pull up the amount of cycles rented per hour.
7. Yulu can offer customised referral discounts based on the bikes they choose in their first trial (Yulu Miracle, Yulu Dex, Yulu Move).
8. The counted vehicle demand on non-working and holiday days is very low. Yulu can introduce promotional offers on holidays and non-holidays or can arrange health competitions on Yulu Move bikes.
9. In summer and fall seasons the company should have more bikes in stock to be rented. Because the demand in these seasons is higher as compared to other seasons.